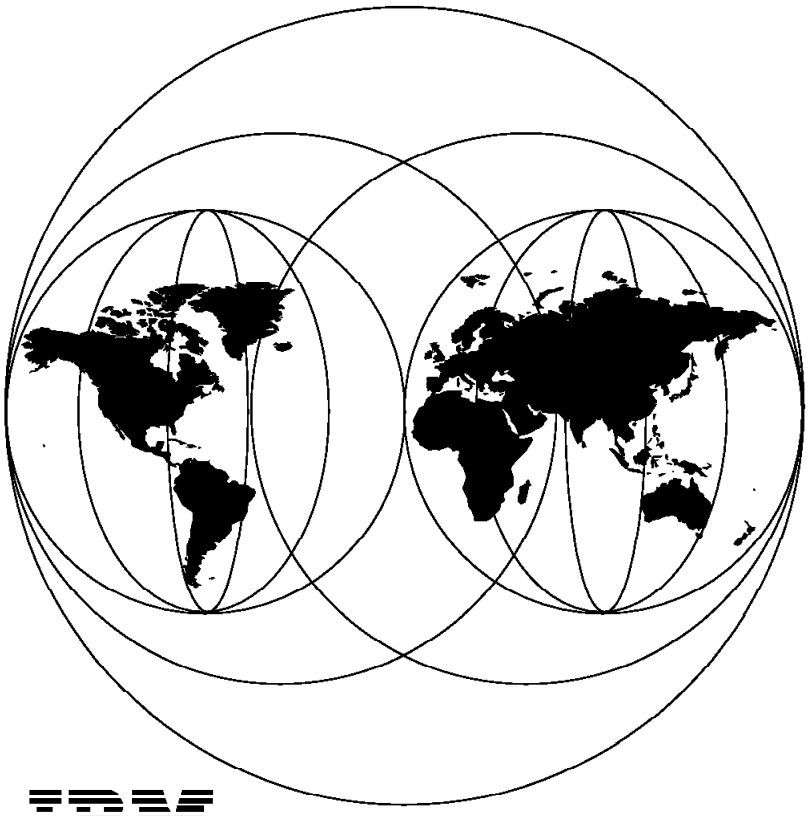
International Technical Support Organization

GG24-2526-00

MVS/ESA OpenEdition DCE: RACF and DCE Security Interoperation

April 1995



IBM

International Technical Support Organization
Poughkeepsie Center

International Technical Support Organization

GG24-2526-00

MVS/ESA OpenEdition DCE: RACF and DCE Security Interoperation

April 1995

Take Note!

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xi.

First Edition (April 1995)

This edition applies to OpenEdition Version 2 DCE Base Services of Program Number 5655-068 and 5655-069 for use with the MVS/ESA SP Version 5 Release 1.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 541 Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

Abstract

This redbook provides an implementation of the design for RACF and DCE security interoperation proposed in the International Technical Support Organization publication, *GG24-4481 MVS/ESA OpenEdition DCE: Application Development Cookbook*. The purpose of the redbook is to help expedite the acceptance of MVS/ESA OpenEdition by demonstrating the use of OpenEdition, including both POSIX and DCE. This is a guide for the MVS programmer, with perhaps a strong systems orientation, into the new world of OpenEdition. The redbook shows how DCE security can be used with RACF security in an MVS environment.

(254 pages)

Contents

Figures

Special Notices

This publication is intended to help IBM customers and system engineers to develop DCE applications on MVS/ESA. The information in this publication is not intended as the specification of any programming interfaces that are provided by MVS/ESA OpenEdition DCE Base Service. See the PUBLICATIONS section of the IBM Programming Announcement for *OpenEdition Distributed Computing Environment for MVS/ESA* (294-501) for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

You can reproduce a page in this document as a transparency, if that page has the copyright notice on it. The copyright notice must appear on each page being reproduced.

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

AIX
CICS
CICS/ESA
Enterprise Systems Architecture/390
ES/9000
IBM
IMS/ESA
MVS/ESA
OpenEdition
S/390
System/390

The following terms are trademarks of other companies:

Open Software Foundation and OSF are trademarks of the Open Software Foundation, Inc.
X/Open and UNIX are trademarks of the X/Open Company Limited
POSIX is a trademark of the Institute of Electrical and Electronic Engineers, Inc.
Windows is a trademark of Microsoft Corporation

Other trademarks are trademarks of their respective companies.

Preface

This document is intended to be used by IBM customers and system engineers that are interested in developing DCE applications on MVS/ESA. Its purpose is to show a person with an MVS/ESA background how to get a distributed application to work in an MVS/ESA environment. It is a “cookbook” showing every step needed to enable RACF and the DCE Security Server to interoperate. With the use of this software an application developer is able to write MVS DCE clients and servers that utilize the security features of OSF DCE and RACF in an integrated way.

This book implements a design originally included in an International Technical Support Organization publication, *GG24-4481-00 MVS/ESA OpenEdition DCE: Application Development Cookbook*. That publication documented a design that would allow RACF and DCE security to interoperate. There have been several minor corrections and improvements to that design. A summary of the changes to the design are included in Summary of Design Changes.

This book assumes the reader is familiar with OSF DCE, RACF, and MVS/ESA. There is no attempt to discuss the fundamentals of DCE, RACF, or MVS/ESA. Please check the “Related Publications” on page xv for further information.

A diskette accompanies this book, and it contains the source code for all the programs listed in the appendix. There are instructions in this book to upload the code to an MVS system.

The programs in this book were developed and tested using an ECIP II system. Early Customer Involvement Program (ECIP) was part of the IBM quality program for MVS/ESA OpenEdition. Its purpose was to get customer feedback on the product early enough in the development cycle to fix problems before they got wide distribution. ECIP I started in early 1993 and provided about a dozen customers with very early version of MVS/ESA DCE. The ECIP II expanded the customer set. It started the summer of 1994 and continued to provide early versions of the OpenEdition product to customers. Although we expect little change in the delivered product from ECIP II, readers should be warned that there may be some differences from what is described in this book.

If at some future date IBM provides a release of OpenEdition and RACF that integrates DCE and RACF security, all efforts will be made to conform to the interfaces presented in this redbook. There are, however, no guarantees that the interface will not change.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “Design”

  This chapter describes a design of a set of programs that will enable RACF and the DCE Security Server to interoperate.

- Chapter 2, “Installation”

  This provides installation instructions of the programs.

- Chapter 3, “Using DCERACF Subroutine and the Utilities”

This chapter describes the use of the utilities associated with RACF and DCE Security Server interoperation.

- Appendix A, “File and Data Set descriptions”

  This appendix contains a list of the contents of the diskette that accompanies this book and a list of the data sets created during the installation process.

- Appendix B, “Source Code”

  This appendix contains a listing of all of the source code.

- Appendix C, “Structured Programming Macro Reference”

  This appendix contains a description of the structured macros used in the assembly language routines.

Summary of Design Changes

This book provides a set of software that allows RACF and DCE security to interoperate. The design was first presented in *GG24-4481, MVS/ESA OpenEdition DCE: Application Developement Cookbook*. Since that design was published a number of changes have been made. We summarize those changes in this section.

Cross Reference Profiles

The original design searched the clients PAC group list to find the RACF userid from a group with RACF_<sysid>_ prefix. When it was found, the userid was extracted from the group name RACF_<sysid>_<userid>. This meant that the DCE Security server had to be queried for every group UUID in the PAC to convert it to its name. Given the time delays across the network and the number of groups involved, it caused a performance problem.

The new design takes the principal UUID from the PAC, converts it to a character string, and uses it as a key to find a RACF general resource profile (referred in this document as the “cross reference profile”) in which the RACF userid in stored in the INSTDATA field.

To support that design, a new utility (DCERACFU) has been created to take a DCE principal name, convert it to an UUID string, and create the “cross reference RACF profile” in which the RACF userid is stored.

Use of USRDATA Instead of INSTDATA Fields

The original design used the user’s RACF user profile INSTDATA field to store the DCE principal name, the encrypted DCE password, and the version of the key used to encrypt the password. It used a / character to delimit the fields in the INSTDATA. However, the / is a valid character to be in the principal name, therefore it was not acceptable.

The new design uses the USRDATA field instead of the widely used INSTDATA (which can be easily changed by the user and RACF administrators) and now uses three offset bytes at the beginning of the USRDATA to the fields.

The above mentioned utility (DCERACFU) enables the USRDATA fields to be set and listed in the user’s RACF profile.

Removal of an Integrity Exposure

A RACHECK post-processing exit has been added. It is used here to close a security hole introduced by the earlier design.

The user SVC described in GG24-4481 assumed that the binding handle for the client session was opaque, meaning that the contents were secured (that is kept encrypted) by DCE runtime. This assumption was incorrect.

Opaque only means that there is no mapping provided for the structure. The contents have already been decrypted by the DCE runtime and it could be modified by a rogue non-APF authorized server.

For example, the binding handle could be searched for the client′s UUID and replaced with another UUID. This UUID is used by the user SVC to logon the client to RACF without a password because the user has already been authenticated by DCE security. That way the rogue server could get access to resources to which the replaced client had access.

To close the integrity hole for a non-APF authorized DCE server, a check is made to verify the security environment matches the address space security environment. This means that the client can only have access to resources to which the DCE application server also has access. While this is a restriction, it still allows the DCE server on MVS to discriminate between clients for the resources to which the server does have access while retaining RACF logging and auditing controls.

Passticket Support

To remove the above restrictions, a DCE application server can choose to use the RACF passticket support. Because the client is fully authenticated by the RACF passticket, the user can be given access to a RACF resource independent of the level of access that the DCE application server may have. For example, an application could create a MVS data set and add or update records where the data set name is not known by the application in advance and, therefore, cannot be added to the access list in anticipation of the request.

To provide this support, the following has been added to the design:

- A passticket server, that will generate a RACF passticket for the DCE client′s associated RACF userid,
- a function to bind to that server
- A function to request a RACF passticket as a client to the above server
- An option to the RACF_logon function of the user SVC to use the passticket as a RACF one-time-use password

---

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *Developing DCE Applications for AIX, OS/2, and Windows*, GG24-4090-01
- *Distributed Computing Environment Understanding the Concepts*, GC09-1478
- *Introduction OpenEdition MVS*, GC23-3010

- *MVS/ESA OpenEdition Application Programmers Guide*, SC23-3017
- *MVS/ESA Client/Server Presentation Guide*, GG24-3931
- *MVS/ESA OpenEdition DCE: Installation and Configuration*, GG24-4480
- *MVS/ESA OpenEdition DCE: Application Support Server for IMS and CICS*, GG24-4481
- *MVS/ESA OpenEdition DCE: Configuration and Getting Started*, SC09-1490
- *MVS/ESA OpenEdition DCE: Application Development Guide*, SC09-1484
- *MVS/ESA OpenEdition DCE: Application Development Reference*, SC09-1487
- *MVS/ESA OpenEdition DCE: Administration Guide*, SC09-1485
- *MVS/ESA OpenEdition DCE: Administration Reference*, SC09-1486
- *MVS/ESA OpenEdition DCE: Configuration and Getting Started*, SC09-1490
- *MVS/ESA OpenEdition DCE: Master Index*, SC09-1488
- *MVS/ESA OpenEdition DCE: Messages and Codes*, SC09-1483
- *MVS/ESA OpenEdition DCE: Planning*, SC09-1489
- *MVS/ESA OpenEdition DCE: Presentation Guide Volume I*, GG24-4240
- *MVS/ESA OpenEdition DCE: User's Guide*, SC09-1704
- *MVS/ESA OpenEdition Release 2 Technical Overview*, GG24-4095
- *MVS/ESA Support for IEEE POSIX Standards Technical Presentation Guide*, GG24-3867
- *SAA AD/Cycle C/370 General Information*, GC09-1358
- *SAA AD/Cycle Language Environment Concepts*, GC26-4531
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *Open Software Foundation DCE Global Directory Services and Office Vision/MVS Enterprise Address Book Overview and Comparison*, GG24-3810
- *RACF V1 R9.2 SPL: RACF*, SC28-1343
- *RACF V1 R9.2 Macros and Interfaces*, SC28-1345
- *RACF V1 R9.2 RACROUTE Macro Reference*, GC28-1366
- *RACF V1 R9.2 Security Administrator's Guide*, SC28-1340
- *RACF V1 R9.2 Command Language Reference*, SC28-0733
- *The RACF Syntax Booklet*, SX22-0014
- *RACF V1 R9.2 Messages and Codes*, SC38-1014
- *RACF V1 R9.2 Auditor's Guide*, SC28-1342
- *RACF V1 R9.2 General User's Guide*, SC28-1341
- *RACF V1 R9.2 Master Index*, GC28-1035
- *RACF V1 R9.2 Program Directory for VM Installations*, GC28-1034
- *RACF V1 R9.2 Program Directory for MVS Installations*, GC28-1054
- *RACF V1 R9.2 Migration and Planning*, GC23-3054

International Technical Support Organization Publications

- *MVS/ESA OpenEdition DCE: Application Development Cookbook*, GG24-4481

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

International Technical Support Organization Bibliography of Redbooks, GG24-3070.

To get a catalog of ITSO technical publications (known as “redbooks”), VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Technical Publications

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their local IBM office.

Customers may order hardcopy ITSO books individually or in customized sets, called GBOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain books on a variety of products.

Acknowledgments

This project was designed and managed by:

Bob Dahle
International Technical Support Organization, Poughkeepsie Center

The author of this document is:

Eric Finkelstein
Australian Programming Centre, ISSC Australia
vnet: ERICFINK at SYDVM1, internet: eric_finkelstein@au.ibm.com

This publication is the result of a residency conducted at the International Technical Support Organization, Poughkeepsie Center.

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Bob Dahle
International Technical Support Organization, Poughkeepsie Center

Rich Guski
RACF Design, IBM US

Chapter 1. Design

1.1 DCE Security and RACF

DCE provides adequate security that is able to be extended to cover all aspects of an application's security.

In an MVS system, security is provided by RACF or its equivalent. How does DCE security and RACF relate to each other? Can they coexist? Can they complement each other?

RACF and DCE security haven't been integrated yet. That integration is to be delivered later. In the meantime, we must deliver solutions that do not compromise the security of the MVS, or DCE and are consistent with the direction of the likely implementation of that integration.

In order to use RACF and DCE security together the following problems have to be solved:

- *Mapping DCE principals to RACF userids*. A DCE principal name can be up to 128 characters. A RACF userid can be up to 8 characters.
- *Mapping DCE authorities to RACF authorities*
- *Implied and explicit login to DCE or RACF* when a user enters the system from either MVS or DCE.
- *Storing of encrypted DCE passwords* to enable implied login.
- *RACF requires that a program wanting to use privileged services, such as RACROUTE ENVIR=CREATE to logon, must be MVS Authorized Program Facility (APF) authorized*
- *Security of MVS HFS files*
- *Access to DCE servers from CICS applications*

The following sections address these issues from two perspectives and provide an implementation of the techniques described:

- A DCE application server on MVS
- A DCE client on MVS

Note: The programs described here are distributed with this book.

Table 1 on page 2 makes a comparison between RACF facilities and the equivalent facility in DCE security.

Table 1. Comparison of RACF and DCE Security

	RACF	**DCE**
Users	Defined as a user profile. Maximum userid length is 8. Maximum password length is 8. Password rules apply.	Defined in DCE registry database as a principal. Maximum principal name length is 1023. Maximum password length is 1023.
Groups	Defined as a group profile. Maximum groupid length is 8	Defined in DCE registry database as a group. Maximum group name length is 1023.
Organization	Not applicable	Defined in DCE registry database as a organization. Maximum organization name length is 1023.
Dataset/file access	Defined by an explicit or generic dataset profile. Maximum dataset name length is 44. It is checked for access authority on data set open by MVS data management and RACF, transparent to the application. Access is controlled by an access list in the dataset profile containing users and groups and the associated access level.	Undefined when DCE/DFS is not used. A DCE application server can use the CDS name space and define a object that represents the dataset/file. It has to be explicitly checked by the application server code using DCE ACLs associated with the object. They contain a list of users and groups and the associated permission bits.
Program/transaction execution	A program in a private library is controlled by a general resource profile in the PROGRAM class and execute access level in the dataset profile so that a program can be executed but not read. Transactions in CICS and IMS also use general resources in the appropriate class to determine whether a user is allowed to execute the transaction.	A server can be controlled by the ACL in the CDS name space that contains the exported bindings of the server. If a client does not have read access to the exported bindings of the server, he or she cannot connect to that server.
General resource	An application can define an application specific class and profiles in that class that represent a logical function in the program.	A server can use an object in the CDS name space to represent a logical function in the program and can use the CDS′s ACL manager to manage the meanings of the permission bits. A server can provide its own ACL manager to make the meanings of the permission bits specific to the application. However, the server has to be executing when the acl_edit utility is executed to change the ACLs of the objects.

1.2 A DCE Application Server on MVS

There are several reasons that a DCE application server would be installed on an MVS system:

- To use a highly available processor with adequate processing capacity
- To use data, such as databases and MVS data sets, stored and managed in the MVS system,
- To use existing transaction processing systems such as CICS, IMS, APPC/MVS servers, and Message Queueing

This section addresses the use of data stored and managed in the MVS system as follows:

- DCE application server on MVS using DCE security control
- DCE application server on MVS using RACF control

Figure 1 illustrates the model we will use in developing the security design. We have a DCE authenticated client communicating to a DCE server using remote procedure calls (rpc). The DCE server running on MVS is subject to RACF (or its equivalent) security controls.

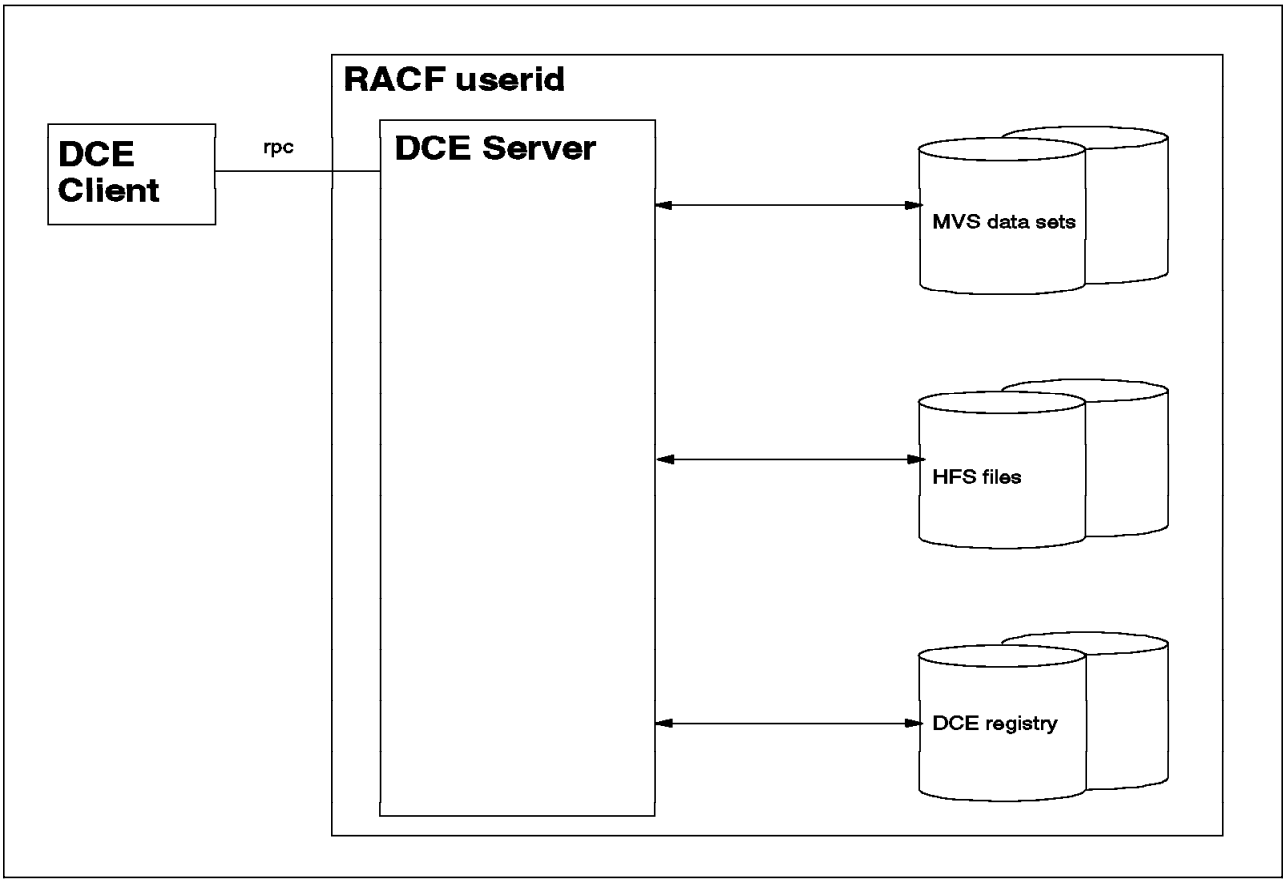


Figure 1. Security Controlled by DCE Security

1.2.1 DCE Application Server on MVS Using DCE Security Control

Appearance of a DCE server to RACF

To RACF, a DCE application server appears as an ordinary RACF user, and is restricted to those MVS resources to which that RACF user has access. Those MVS resources may be data sets, or general resources (such as tape volumes, or application defined resources).

DCE security differs from the RACF philosophy in that the implementation of protection is built into the design of the server rather than being external to the application. RACF can control whether a user is allowed to execute a program or gain access to a data set and at what level from the RACF profiles external to

the application. RACF can also enable an application to define resources that can represent processes within the application through RACF general resources.

Access to MVS/ESA OpenEdition DCE HFS files

A DCE application server can make use of files inside the HFS file system, and MVS data sets. By default, the HFS file system provides the local file system for MVS/ESA OpenEdition DCE.

Each HFS file set is implemented as a new MVS file type of HFS. Multiple file sets are mounted to form a tree structure of files similar to that in use on UNIX and OS/2 systems. The control of access to files within the HFS file system is based on permission bits as specified in POSIX 1003.1.

Permission bits are associated with each file and are as follows:

Owner permissions - rwx (read, write, execute)

Group permissions - rwx

Other permissions - rwx

The “owner” permissions apply to the person that created the file. “Group” permissions apply to the group to which the owner belongs. In a RACF system, each RACF group is assigned a group ID and the group ID recorded for the file is the owner′s default group. The “other” permissions apply to anyone else.

To DCE security, the HFS files are files in the local file system. The next section discusses MVS data sets and applies to HFS files as well.

Access to MVS data sets

To DCE security, MVS data sets (and HFS files) are considered to be files in another local file system, and are outside the scope of DCE security control. However, the manner of use of these local files that the application has access to within the local file system can be controlled by the DCE ACLs that represent that use.

A DCE application server is expected to use the DCE security to determine whether its client is:

- Authenticated
- Granted permission by name
- Granted permission by being a member of a group
- Granted permission by the ACL representing a resource and at what access level

Access to RACF general resources

Any RACF application can use RACF general resources as a part of its processing to represent a logical function or access to portions of data sets/files to which the application has access. However, even though the DCE clients are authenticated and their identity can be trusted, they are not known to RACF. All access checking in RACF is done based on the DCE server′s RACF userid.

Because a normal DCE server is running in MVS problem program state, that is APF unauthorized, it is not allowed to issue the privileged “RACROUTE

ENVIR=CREATE" request to identify a user to RACF, so that RACF checking will be based on the DCE server instead of the client.

The implication is that, if an application needs to create a RACF environment for a client, it will need to:

- Execute APF authorized, and
- Have some means of mapping a DCE principal to a RACF userid.

In most MVS installations, running applications APF authorized is not normally allowed, and if it is absolutely needed, it must be strictly controlled.

In section 1.2.2, "DCE Application Server on MVS Using RACF Control," these problems are discussed and possible solutions are provided.

1.2.2 DCE Application Server on MVS Using RACF Control

In a typical MVS installation, there is already a substantial user population that is governed by security policies implemented through RACF profiles. The introduction of DCE into these systems presents the installation with a need to implement their security in DCE as well as RACF to protect their systems, and maintain those policies in both environments. This presents a RACF security auditor the need to inspect and certify DCE application servers to see that the policies have been implemented correctly.

In this section, we describe a technique that enables a DCE application server the use of RACF security based on a DCE principal's associated RACF userid without the DCE application server having to be an MVS authorized program (APF).

An MVS APF authorized program has to reside in extremely well protected, nominated, MVS libraries who's contents are monitored. This is because an MVS APF authorized program can put itself into supervisor state, change its storage key or do anything it wants in the MVS system, intentionally or unintentionally, including bypassing any security.

To remove the potential of unacceptable exposure of the MVS system, the following design introduces an MVS user SVC (supervisor call) (see appendix B.2.6, "USERSVC" on page 81 for its implementation).

An SVC is given control in supervisor state, storage protection key zero. Within that routine, it can issue the privileged parameters of the RACROUTE to, for example, logon a user onto RACF. To logon to RACF, however, the SVC has to have a means of finding a DCE client principal's associated RACF userid.

To map from a DCE principal name to a RACF userid, we use the DCE PAC (privileged attribute certificate) that accompanies an authenticated DCE RPC. The PAC contains the UUIDs of the principal and the list of DCE groups to which the principal belongs.

The DCE principal UUID is converted into a character string, and using it as the RACF entity name, a previously created RACF general resource profile is read. This profile has the associated RACF userid recorded in its INSTDATA field.

Note: *These profiles will be called UUID cross reference profiles in this document. They provide a cross-reference between the DCE principal and the user's RACF userid. These profiles are created by a utility, DCERACFU,*

distributed with this book and described later (see section 1.4.2, "DCERACFU Utility" on page 17).

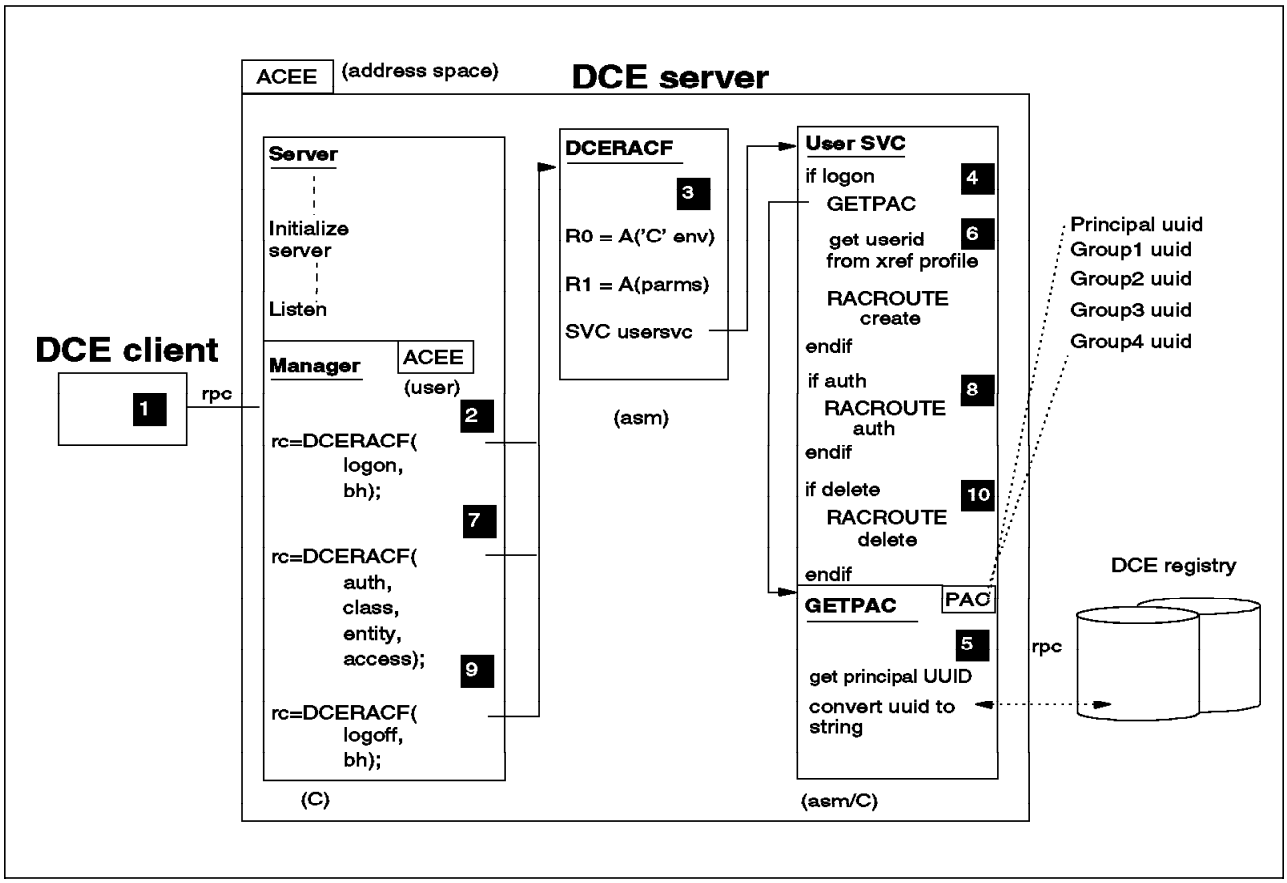


Figure 2. Security Controlled by RACF

The numbered points in Figure 2, are discussed in the following steps, which show the logic flow for establishing a RACF environment for a DCE application server.

1. An authenticated DCE client of a DCE application server issues an **rpc_binding_set_auth_info** request to indicate to the DCE runtime the name of the server, and the authentication service to be used. When the authentication service requested is **rpc_c_authn_dce_secret** or **rpc_c_authn_default**, a PAC (privilege attribute certificate) is built and forwarded with every RPC issued. The RPCs are called authenticated RPCs. The PAC contains the UUIDs of the principal and a list of groups that the principal is a member.

2. The DCE application server calls an assembler subroutine, DCERACF (which calls the user SVC), to create a RACF ACEE for the client's RACF userid.

   Because DCE security provides third party authentication of the client principal, we can treat it as a trusted peer and do not need to provide the RACF password.

   The ACEE is chained from the MVS TCB, and all subsequent requests to RACF through the RACROUTE macro for authorization or implied by the use of MVS services will be based on the client's RACF userid.

   We do not rely on parameters passed by the caller of subroutine apart from the RPC's binding handle. All other information needed, such as the client's

authentication level, certification, and the client's PAC, are requested directly by the user SVC (through a C subroutine). In this way the use of the SVC is restricted to only being able to logon a DCE client's associated RACF userid [1].

The DCERACF subroutine parameters depend on the type of call. To logon to RACF, and create an ACEE for a user, the parameters are as follows:

```
rc=DCERACF("RACF_logon", <binding_handle>);
```

where:

RACF_logon requests a logon to RACF.

<binding handle> is the handle passed from DCE runtime for this RPC.

rc is the return code of:

0	RACF logon OK, ACEE created
4	RACF userid has been revoked
8	invalid password or RACF userid not known to RACF
12	client is not using an authenticated RPC
16	client does not have a UUID cross reference profile from which the RACF userid is extracted.

3. The DCERACF subroutine is a C conformant assembler routine that issues the user SVC, passing on the caller's parameters, and the C enclave environment registers.
4. For a RACF_logon request, the user SVC has to re-establish the inherited C language enclave from the original caller, and calls the C function, GETPAC.
5. The GETPAC function (see appendix B.3.2, "GETPAC" on page 134) is passed the binding handle. Using that binding handle, it can make requests to DCE to determine details about the client principal. To protect the integrity of MVS, we do not use any information provided by the caller of the SVC apart from the binding handle, which is verified by DCE.

   GETPAC returns the DCE client principal UUID string to the user SVC, as follows:

   a. It uses **rpc_binding_inq_auth_client** to get the client's credentials in the PAC (see "rpc_binding_inq_auth_client" on page 135). The PAC contains the UUID of the client principal and a list of UUIDs of groups of which the principal is a member. The PAC is provided by the DCE runtime and was part of the original RPC call from the client.

   b. It uses **uuid_to_string** to convert the binary UUID into a 36 character UUID string (see "uuid_to_string" on page 136).
6. On return from the GETPAC function, the user SVC will:

   a. Use RACROUTE TYPE=EXTRACT to read the INSTDATA field of the UUID cross reference profile, from which it gets the DCE client's associated RACF userid (see "GETUUID subroutine, Extract INSTDATA field" on page 112).

---

[1] Because the binding handle may be tampered with by a "rogue" server, a restriction is enforced on an non-APF authorized DCE application server, requiring that the client's access to RACF resources cannot be greater than the application server's access to the resource. If the application chooses to use the passticket server (see 1.4.4, "Passticket Support" on page 22), these restrictions can be removed.

b. Use RACROUTE REQUEST=VERIFY,ENVR=CREATE to create the ACEE (see ″LOGON subroutine, Issue RACROUTE″ on page 87). The SVC can use the NOPASSWD parameter because the user has been authenticated by a trusted system, DCE. The ACEE is chained off of the caller′s TCB (the manager).

   To enable the RACF RACHECK post-processing, ICHRCX02, to recognize an ACEE that has been created by this user SVC, the APPL parameter has been set to “DCERACF.” If, however, the passticket server is used (see section 1.4.4, “Passticket Support” on page 22), the APPL parameter is set to DCE<smfid> and there are no restrictions placed on the client′s normal access.

7. Once the RACF environment has been established, all access to MVS resources is determined by the RACF userid or the RACF groups to which the client belongs. Opening a MVS data set, or allocating a new MVS data set will be allowed or denied through requests by the MVS data management routines to RACF.

   If the DCE application server wants to explicitly check the authority to use a RACF resource (dataset, tapevol, general resources, and so on), the parameters are as follows:

```
rc=DCERACF(″RACF_check_authorization″,
           <class>, <entity>, <access requ  ested>);
```

   where:

   RACF_check_authorization requests an authority check

   <class> is the class of RACF entity (for example, DATASET)

   <entity> is the RACF entity being requested

   <access requested> is the access requested (for example, read, write, control, alter).

   rc is the return code of:

0	access granted
8	access denied
12	resource not known
16	Don′t know

8. For an check_authorization request, the DCERACF assembler subroutine issues the user SVC [2] which issues a RACROUTE REQUEST=AUTH using the parameters passed from the caller (see ″CHKAUTH subroutine, Issue RACROUTE AUTH″ on page 92).

---

[2] The RACHECK post-processing exit (see appendix B.2.3, “ICHRCX02” on page 72) has been added to the design. It is used here to close a security hole introduced by the original design.

The user SVC provided assumed that the binding handle for the client session was opaque, meaning that the contents were secured (that is kept encrypted) by DCE runtime. This assumption was incorrect.

Opaque only means that there is no mapping provided for the structure. The contents have already been decrypted by the DCE runtime and it could be modified by a rogue non-APF authorized server.

For example, it could be searched for the client′s UUID and replaced with another UUID. This UUID is used by the user SVC to logon the client to RACF without a password because the user has already been authenticated by DCE security. That way the rogue server could get access to resources to which the client had access without haveing access itself.

To close the integrity hole for a non-APF authorized DCE server, this exit forces any RACHECK for an ACEE pointed to by TCBSENV, which has the ACEEAPLN field in the ACEE containing “DCERACF,” to be re-driven to also check against the

9. The DCE application server must call the subroutine DCERACF to delete the ACEE for user before completing processing for this client. The parameters are as follows:

   ```
   rc=DCERACF("RACF_logoff", <binding_handle>);
   ```

   where:

   RACF_logoff requests deletion of the ACEE.

   <binding_handle> is the DCE server's binding handle

   rc is the return code of:

   0 logoff successful

   8 not logged on

10. DCERACF, issues the user SVC requesting that the RACF userid be logged off. The user SVC issues a RACROUTE REQUEST=VERIFY,ENVR=DELETE (see "LOGOFF subroutine, Issue RACROUTE DELETE" on page 93).

1.3 A DCE Client on MVS

The following lists the probable environments in which DCE clients can work, or would like to work:

- TSO client
- OMVS client
- Batch client
- CICS client
- IMS client
- APPC/MVS client

TSO, OMVS, and batch environments are the most readily useable for a DCE client. A TSO, or OMVS user logs onto TSO and then issues a DCELOGIN (or DCE_LOGIN) to which he or she provides his or her DCE password in the clear. His or her DCE login context is created and held in the EUVSKRB5 cache data set. Likewise, a batch job is submitted from TSO, and one step executes the DCELOGIN passing the principal and DCE password in the PARM parameter.

However, what about a TSO user who executes a program that becomes a DCE client under the covers? Do all DCE client applications have to assume that the user has logged in to DCE?

---

address space ASXBSENV ACEE when the TCBSENV ACEE RACHECK was successful.

This means that the client can only have access to resources to which the DCE application server also has access.

While this is a restriction, it still allows the DCE server on MVS to discriminate between clients for the resources to which the server does have access while retaining the RACF logging and auditing controls.

Note: If the installation already uses the ICHRCX02 exit, this code will have to be merged with it to ensure the integrity exposure is closed. This code will only process the request if the field ACEEAPLN (the application name) in the ACEE is DCERACF and the job step is non-APF authorized. Therefore, only ACEEs created by the user SVC will be affected by this code.

Note: The restrictions placed through the ICHRCX02 exit, described above, can be removed if the DCE application chooses to use the passticket server (see 1.4.4, "Passticket Support" on page 22) and the client becomes a fully authenticated RACF user. For a passticket RACF logon, the ACEEAPLN in the ACEE is DCE<smfid>, where <smfid> is the MVS system's SMF system identification (SID parameter in member SMFPRM00 of SYS1.PARMLIB).

What about a CICS application that needs to become a DCE client to a DCE application on a UNIX system [3]?

CICS has the added complication that it is executed in MVS as a single task (TCB) and provides its own dispatcher. As a result, a CICS transaction can never WAIT on an MVS event or, in DCE terms, block the task until the request has been executed. Such activity would put the whole CICS region into a WAIT at the expense of all other users in that CICS region.

Figure 3 on page 11 describes a technique [4] that uses APPC/MVS as a tailored proxy or gateway that allows a CICS application to become a DCE client to a DCE server that may be on a UNIX system.

The technique is used here to illustrate how an application can:

- Login a DCE user on the basis that his identity has been authenticated by RACF
- Provide the DCELOGIN with a clear password even though the user cannot readily respond to a prompt for the password (the CICS user who is unaware that the CICS transaction is talking to a DCE server)

The technique is applicable to all the DCE client environments described above without the need for the use of APPC/MVS.

[3] This is the converse of the MVS/ESA OpenEdition DCE optional features, CICS and IMS Application Support Servers, which only provide access to legacy transactions in CICS and IMS from DCE clients on the DCE network.

[4] This is not the only technique that could be used with CICS. For example, at initialization of a CICS region, MVS subtasks could be created that could service a queue of requests to become a DCE client and post the CICS transactions upon completion of the request.

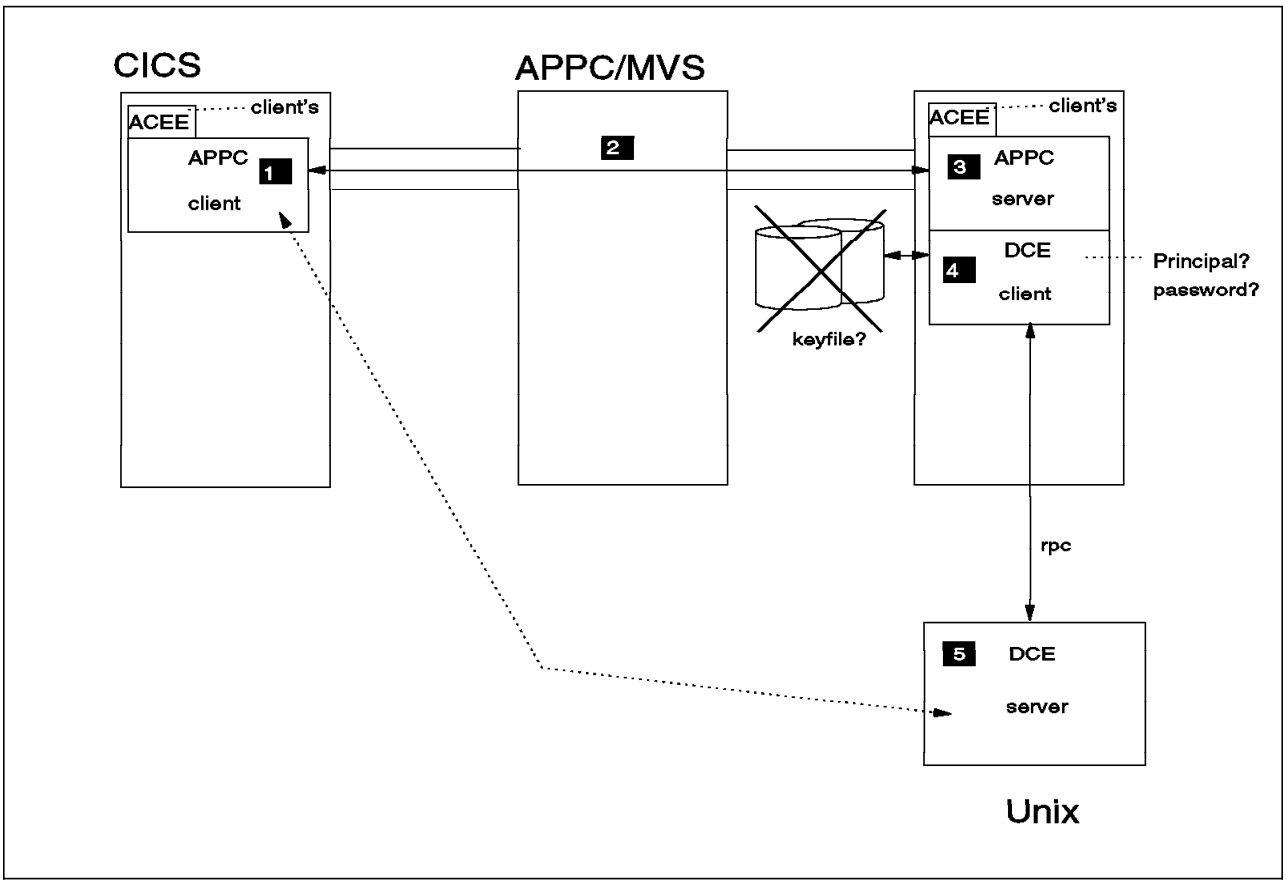


Figure 3. CICS Client and DCE Server

The following points describe Figure 3:

1. The intent of the CICS transaction is to become a client to the DCE server shown in the diagram as point 5.

   A CICS user has a RACF ACEE created when he or she logons to CICS. All access to resources, including the execution of transactions, are determined by RACF using the userid in this ACEE. The transaction executes a request to invoke an APPC/MVS transaction program and passes on the authenticated userid.

2. During the initialization of CICS, the connection between CICS and APPC/MVS is established as trusted partners. When APPC/MVS receives the connect request to establish a session, it determines that it is to be scheduled to be executed in the dependent address space, ASCH.

3. The scheduler in the dependent address space, ASCH, issues a MVS RACROUTE to establish the RACF ACEE for the requesting user. The transaction program is executed as a normal non-APF program.

   The transaction program is the APPC server requested by the CICS transaction. The APPC server receives the request's data through one or more receive and sends.

4. The APPC server now login to DCE as a client so that the request is satisfied by the DCE server after which the data is forwarded back to the original CICS transaction.

   Now the problem becomes apparent! What is the DCE principal to be used?

- What is his or her DCE password?
- Can we communicate back to the original CICS transaction to prompt for the principal name and DCE password?
- Do we use the DCE keyfile containing principal names and their encrypted DCE password?
- If we do use a keytab file, how do we associate a RACF userid and a DCE principal name?
- If we do use a keytab file, the RACF userid must have read access to the file and as such has access to the encrypted passwords of all other users of this APPC server.

The suggested solution to all of these questions is described in section 1.3.1, “DCE Client Using DCE Password Stored in RACF Database.” For completeness of this example, we outline the solution here.

a. The client′s RACF user profile contains an installation user data field, USRDATA, which the RACF administrator has entered as:

```
DCE(<principal name>)
```

b. The DCE principal updates this field to enter his or her DCE password, by using the SAVEPW command described in 1.4.3, “SAVEPW Command” on page 20. This command will encrypt the DCE password and insert the version number of the encryption key.

```
DCE(<principal name> <encrypted DCE password> <version number>)
```

c. The routine **DCERACF(get_principal,..)** uses the installation′s user SVC to extract the principal name from the current RACF user profile, USRDATA field.

d. The routine **DCERACF(get_password,..)** uses the installation′s user SVC, to extract the clear DCE password from the current RACF user profile, USRDATA field.

The APPC server can now login to DCE as the nominated principal using his or her password and call the DCE server.

5. The DCE server processes the request according to the client′s PAC and returns the result, and in turn, the APPC server returns the result back to the original CICS transaction.

1.3.1 DCE Client Using DCE Password Stored in RACF Database

As discussed in the previous section, the RACF database is used to store the DCE principal name, his or her encrypted DCE password, and the version of the system encryption key in the RACF user profile user data field. This section and the following sections describe:

1. An assembler routine, **DCERACF(get_principal,..)** that will return the principal name from the current RACF user′s profile, USRDATA field.
2. An assembler routine, **DCERACF(get_password,..)** that will return the clear password from the current RACF user′s profile, USRDATA field.
3. A user SVC that encrypts/decrypts the user′s password in the current RACF user′s profile, USRDATA field.
4. The SAVEPW command (see section 1.4.3, “SAVEPW Command” on page 20) that the DCE principal has to execute to store the encrypted DCE password in the user′s RACF user profile, user data field.

5. The DCEKEY utility (see section 1.4.1, “DCEKEY Utility” on page 15) which creates a system secret key under which the user′s userid and password are encrypted.

The technique described here has made the following assumptions:

- The DCE password and the RACF password may be different.
- This implementation restricts the lengths of the DCE principal and the DCE password to a total length of 248 bytes.

  ```
  length of DCE principal + length of DCE password < 249
  ```

  This limitation comes from the length of the RACF USRDATA field.
- The synchronization of password changes on DCE and RACF is done by the SAVEPW command, which will update both the DCE security server registry password and the RACF user profile with the same value.
- The installation has not used the RACF user profile, USRDATA field for other purposes.

To set up a user:

1. The RACF administrator uses the utility DCERACFU to set the associated DCE principal name in the RACF user′s profile.

   ```
   DCERACFU set racf <DCE principal name> <RACF userid>
   ```

   The DCE “principal name” is included to allow the RACF userid and the DCE principal name to be different.
2. The RACF administrator uses the utility DCERACFU to create a UUID cross reference profile for the DCE principal′s UUID to associate the DCE principal name with its RACF userid.

   ```
   DCERACFU set dce  <DCE principal name> <RACF userid>
   ```
3. The RACF administrator also has to create a new RACF general resource class of $DCERACF. This class is used to store the current system secret encryption key and previous versions of it. These profiles must specify UACC of NONE and have nothing in the access lists. The system secret encryption key will be used to encrypt the userid to form an encryption pad when encrypting/decrypting the user′s password in the user′s RACF user profile (see section 1.4.1, “DCEKEY Utility” on page 15 for more details).
4. The DCE principal has to login to DCE [5] and execute the SAVEPW command (see section 1.4.3, “SAVEPW Command” on page 20), initially or whenever he or she changes the DCE password, to store the encrypted DCE password in the user′s RACF user profile, user data field.

   ```
   SAVEPW <sysid> <old password> <new password> <verify password>
   ```

Figure 4 on page 14 illustrates the process:

[5] Login to DCE can be done in a TSO session, or on a non-MVS system in the DCE cell. For users who do not have access to either, such as CICS and IMS users that have not been given a TSO session by the installation, the DCE security administrator will have to issue the SAVEPW command on their behalf. Once the initial SAVEPW command has been issued, however, a CICS transaction could be written (using APPC/MVS), or an IMS transaction, that will allow the user to login to DCE, change the DCE password, and issue the SAVEPW as a client.

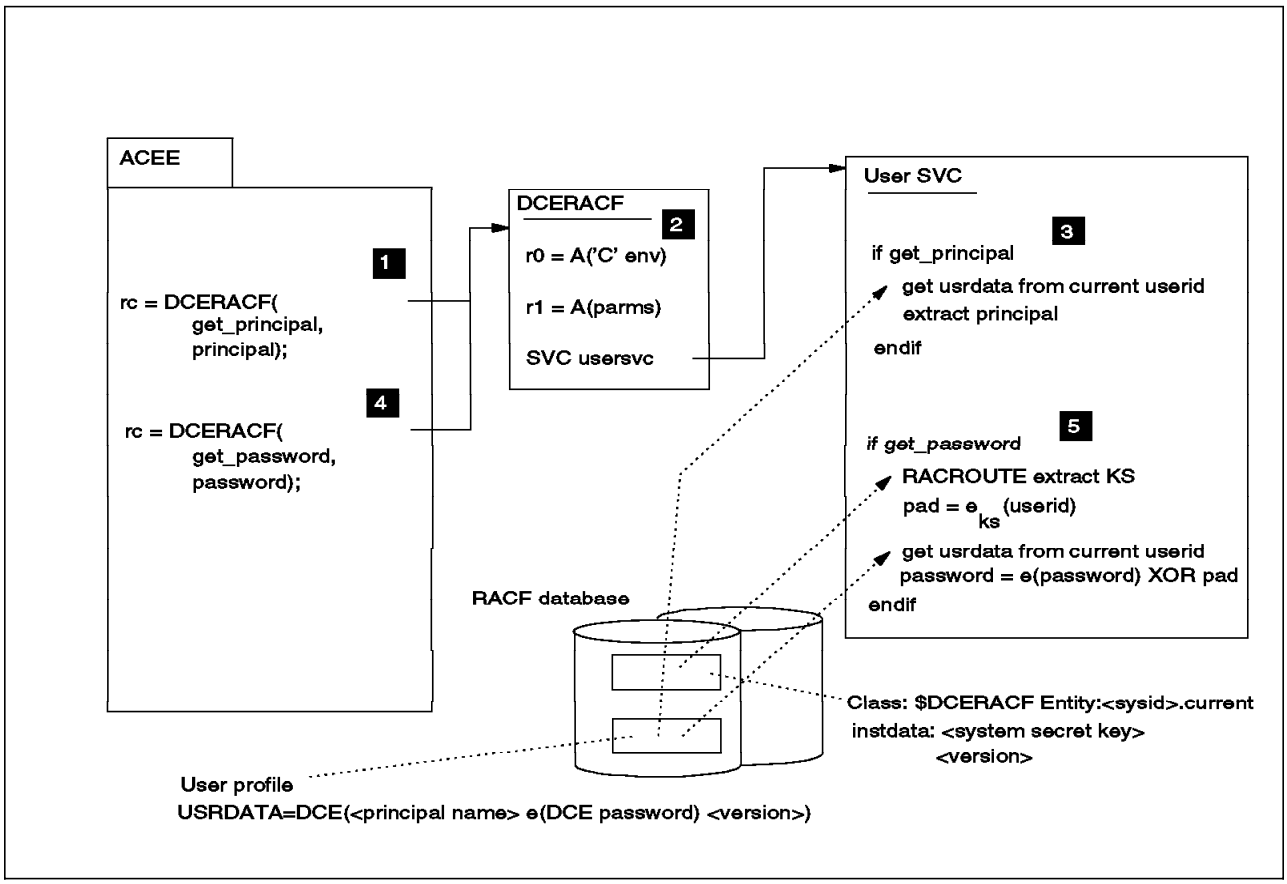


Figure 4. Encrypted DCE Password in RACF Database

1. The application program that wants to issue a login to DCE calls the assembler routine **DCERACF(get_principal, principal)**, passing it a work area that will contain the user's principal name on return. This can be a maximum of 128 bytes.
2. DCERACF calls the user SVC passing the caller's parameters and the C enclave registers.
3. The user SVC extracts the principal name from the current RACF user profile, USRDATA (see "GETPRIN subroutine, Get USRDATA" on page 103).
4. The application program then calls the assembler routine **DCERACF(get_password, password)**, passing it a work area that will contain the user's clear password on return. The DCERACF routine calls the user SVC
5. The user SVC, a fragment of which is shown as pseudo code in the diagram, does the following:
   a. Issue the RACROUTE EXTRACT to retrieve the encrypted DCE password and the version of the system secret encryption key used (see "GETPWD subroutine, Get USRDATA" on page 97). If the current RACF userid does not have the DCE data in the USRDATA field, return to the caller with an error code.
   b. Issue the RACROUTE EXTRACT to retrieve the appropriate version of system secret encryption key (see "GETPWD soubroutine, Extract secret key" on page 99). The <sysid> would be extracted from the MVS CVT.

RACF returns the address of the results in register 1. If the profile is not found, the SVC will return to the caller with an error code.

c. The SVC then issues a RACROUTE TYPE=ENCRYPT to encrypt the current userid in the ACEE under the system secret encryption key to be used as an encryption pad (see ″GETPWD subroutine, Encrypt userid″ on page 100)

d. Because the SVC was requested to decrypt the user′s password, the SVC takes the encrypted password and exclusive OR′s the PAD generated in the previous step with it. This will expose the user′s clear password, and the result is passed back through the calling sequence to the DCE application (see ″GETPWD soubroutine, XOR pad on password″ on page 100).

   Not shown in the diagram for sake of simplicity: if the version of the system secret encryption key does not match the version in the user′s RACF profile user data field, the DCE password that has been decrypted and is in clear, is re-encrypted using the current system secret encryption key, and the user′s RACF profile in the RACF database is updated. In this way, when the system secret encryption key is changed, any reference to a password will automatically re-encrypt it under the new key.

1.4 Utilities

To support the user SVC described above, four utilities are provided:

- **DCEKEY**, which creates a system secret encryption key used to encrypt the DCE client′s password
- **DCERACFU**, which enables the RACF administrator to set the DCE principal name in a RACF user profile, and set the RACF userid in the UUID cross reference profile
- **SAVEPW**, which enables a DCE client to save the encrypted DCE password in his or her RACF profile
- **PTKTS**, which provides support of, and generates, RACF passtickets

1.4.1 DCEKEY Utility

The DCE system secret encryption key utility, DCEKEY, is used to create a secret key to be used to encrypt a user′s password. The utility is APF authorized and has to be placed in an authorized library, so that it can issue privileged parameters on the RACF RACROUTE macro.

The key that is generated is chosen using a random number generator and encrypting that number. That key is stored in the user data field of a RACF general resource in the a new class of $DCERACF. The naming convention for the profiles is:

```
<sysid>.CURRENT | Vnnn
```

where:

<sysid> is the system id as stored in the MVS CVT of this system.

CURRENT or Vnnn identifies the key as being the current key or the version number of the system secret encryption key.

The utility assumes the following:

- The RACF administrator has created a RACF class of $DCERACF.
- The RACF administrator has created a pool of general resource profiles in that class conforming to the naming convention above. The minimum number of profiles is two: one for CURRENT and one for the version number copy of that profile. There should be at least one more version number profile than the number of times the utility has been executed.

Figure 5 illustrates the processing performed by the utility in pseudo code.

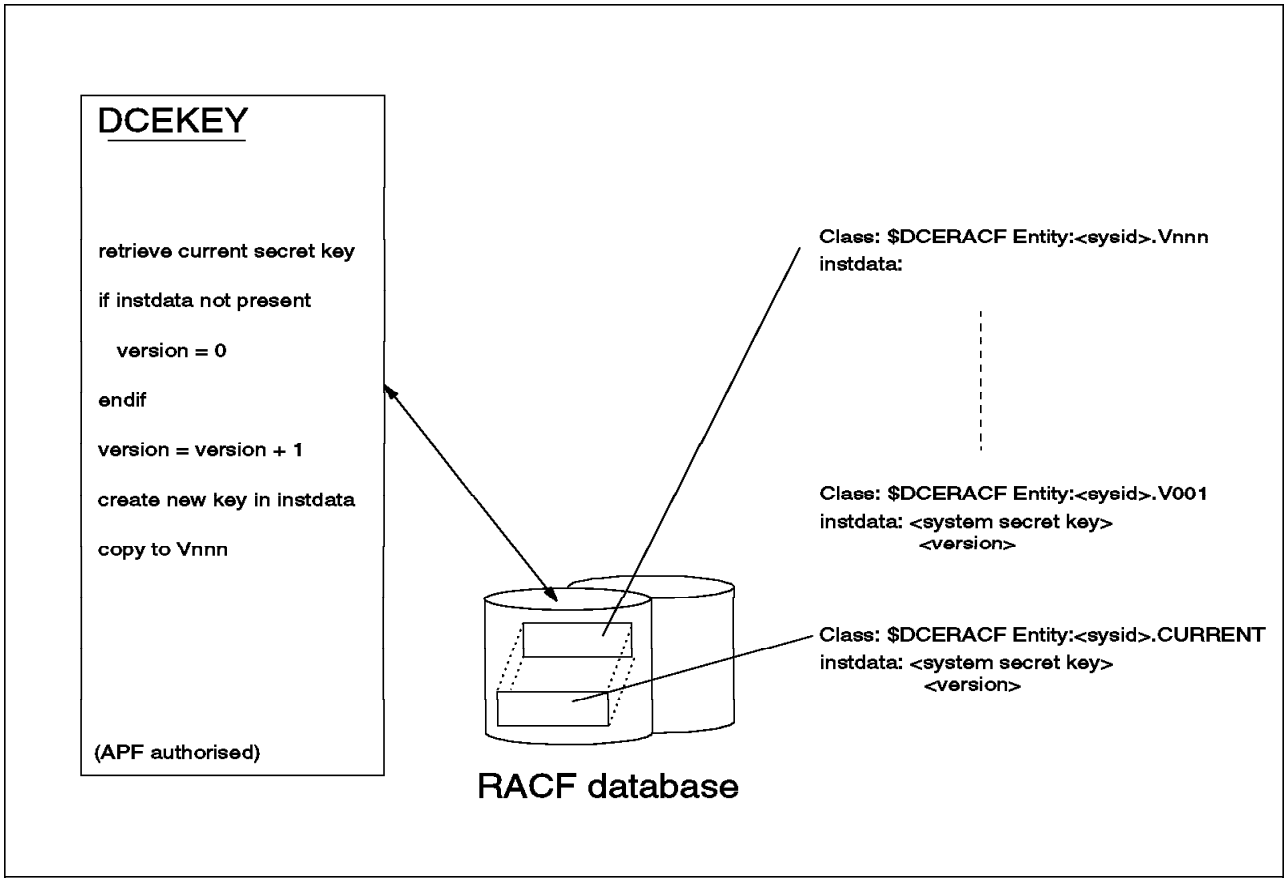


Figure 5. DCEKEY Utility

1. Retrieve the CURRENT profile using RACROUTE REQUEST=EXTRACT to get the key and version number of the key (see ″DCEKEY, retrieve current secret key″ on page 64).
2. If the user data field is not present, set the version number to 0.
3. Increment the version number.
4. Generate a random number and encrypt the system-id with it using RACROUTE TYPE=ENCRYPT (see ″DCEKEY, generate random key″ on page 64).
5. Update the <sysid>.Vnnn profile using RACROUTE REQUEST=EXTRACT, TYPE=REPLACE (see ″DCEKEY, Update Vnnn profile″ on page 65).
6. Update the <sysid>.CURRENT profile using RACROUTE REQUEST=EXTRACT, TYPE=REPLACE (see ″DCEKEY, Update CURRENT profile″ on page 66).

1.4.2 DCERACFU Utility

The DCERACFU utility has several functions.

For an enduser:

- It allows an end user to set his or her DCE password (encrypted) in his or her RACF profile USRDATA field. To ensure that the password known to the DCE security server is the same, the password is also updated in the DCE registry. If the password has never been set, only the RACF USRDATA field is updated.
- It allows an end user to login to DCE using the DCE principal name and the encrypted DCE password in his RACF user profile USRDATA. It can be used in a TSO clist executed automatically when a user logon to TSO and thus giving an effective single network logon.

For an administrator (The administrator must have ALTER access to a RACF general resource ADMINISTRATOR in class $DCERACF):

- It allows an administrator to control who can implicitly login to DCE. The administrator uses this utility to set the associated DCE principal name in the USRDATA field of the nominated RACF user profile.
- It allows an administrator to control who can implicitly logon to RACF. The administrator uses this utility to create a UUID cross reference profile and set the RACF userid associated with a DCE principal
- It allows an administrator to list the DCE segment stored in the USRDATA field of the nominated RACF userid.
- It allows an administrator to list the UUID cross reference profile for a DCE principal.

The parameters to DCERACFU are subcommands that are passed on the command line of the DCERACFU CLIST as follows:

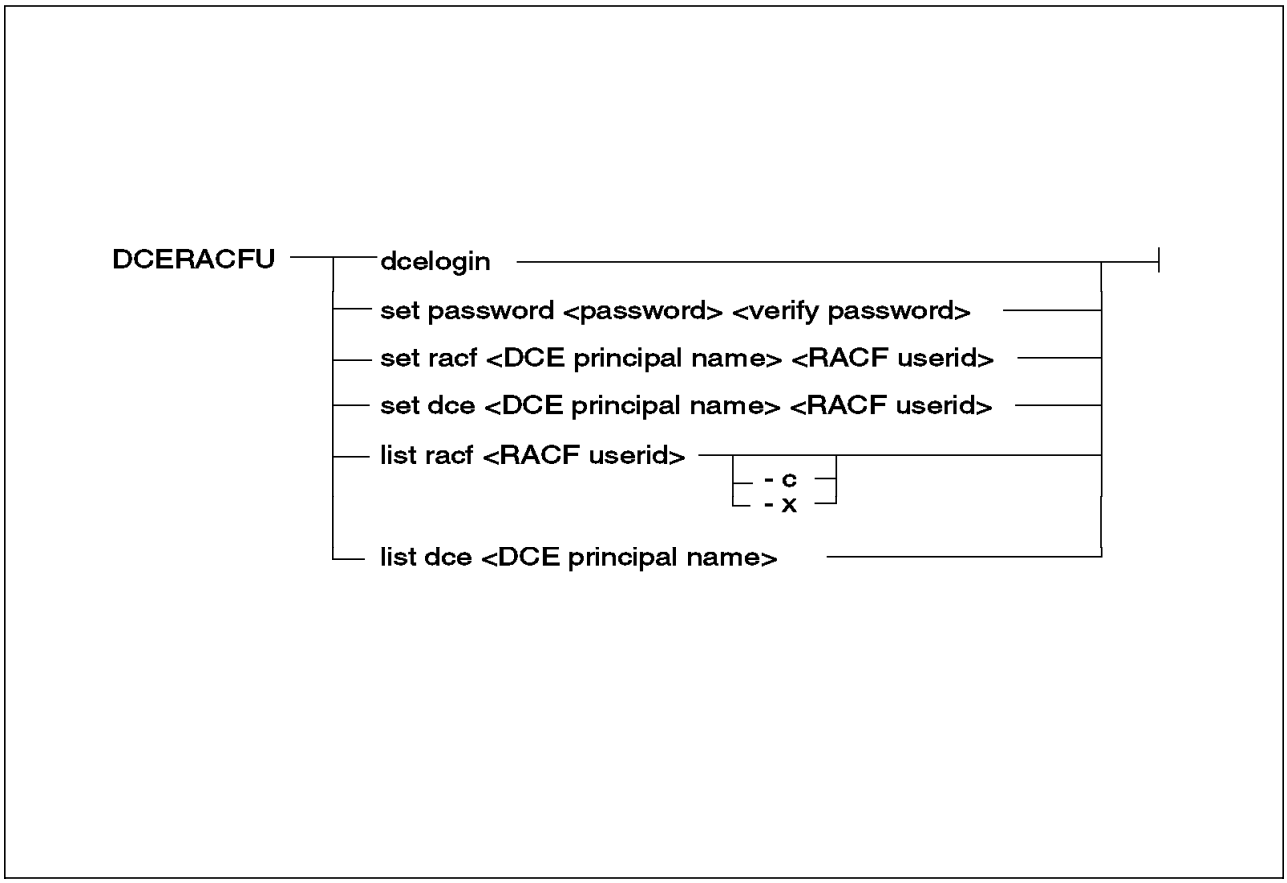


Figure 6. DCERACFU Utility Syntax

The subcommands may also be used in the PARM field in JCL. For example:

```
//jobname JOB ......
// EXEC PGM=DCERACFU,PARM='dcelogin'
//STEPLIB DD DSN=hlq.DCERACF.LOAD,DISP=SHR
```

Note: DCE principal name and password are case sensitive. When editing JCL with the ISPF editor, ensure that CAPS OFF is set.

The processing for each subcommand follows:

DCERACFU dcelogin

dcelogin subcommand (see ″DCERACFU, Function dcelogin″ on page 126) executes a DCE login for the current RACF user, TSO or batch, using the DCE principal name and encrypted DCE password from the RACF user profile, USRDATA field. The user is not prompted for any parameters. dcelogin is intended to be used in an automatically executed CLIST from the TSO LOGON panel.

The dcelogin subcommand:

1. Gets the DCE principal name from the current RACF user profile (see ″DCERACFU, get DCE principal name″ on page 126).
2. Gets the DCE password from the current RACF user profile (see ″DCERACFU, get DCE password″ on page 126).

3. Builds the password structure to be used (see ″DCERACFU, setup password record″ on page 126).

4. Performs the normal DCE login sequence of DCE calls (see pages 127 through 127).

DCERACFU set password <password> <verify password>

This subcommand is an alternative to the use of the SAVEPW command to allow the enduser or the administrator on his or her behalf, to set the initial and subsequent DCE password in the user′s RACF profile. The subcommand works with the currently logged on RACF user. An administrator should use SAVEPW in preference to this subcommand. The subcommand updates the DCE registry as well as the USRDATA field of the RACF user profile if this is not the first time the password is stored in the RACF user profile.

The subcommand:

1. Will attempt to login to DCE (see ″DCERACFU, login to DCE″ on page 128). If this is the first time use for this RACF profile, it will fail because there is no password available yet and therefore the DCE registry cannot be updated.

2. Will open an update DCE registry with the new password (see ″DCERACFU, open update registry site″ on page 128 through ″DCERACFU, unbind registry site″ on page 129)

3. Will update the DCE password in the RACF profile (see ″DCERACFU, DCERACF″ on page 129).

DCERACFU set racf <DCE principal name> <RACF userid>

This subcommand sets the DCE principal name in the RACF user′s profile, USRDATA (see ″DCERACFU, set_racf″ on page 129). This creates the USRDATA field without a DCE password as indicated by a zero offset to the password and encryption key version number. The subcommand uses the DCERACF(RACF_set_DCE_principal,...) function in the user SVC.

DCERACFU set dce <DCE principal name> <RACF userid>

This subcommand sets the RACF userid in the UUID cross reference profile, INSTDATA (see ″DCERACFU, set_dce″ on page 129). The subcommand converts the DCE principal name into a UUID string and the DCERACF(RACF_set_DCE_uuid,...) function in the user SVC to create or overwrite the UUID cross reference profile.

DCERACFU list racf <RACF userid> <-c|-h>

This subcommand enables the administrator to list the contents of the nominated RACF user profile, USRDATA in character or hex form (see ″DCERACFU, list_racf″ on page 132). It uses the DCERACF(RACF_get_DCE_usrdata,..) function of the user SVC.

DCERACFU list dce <DCE principal name>

This subcommand enables the administrator to list the contents of the nominated UUID cross reference profile, INSTDATA (see ″DCERACFU, list_dce″ on page 130). The subcommand converts the DCE principal name into an UUID string and the DCERACF(RACF_get_DCE_uuid,...) function in the user SVC to get the associated RACF userid.

1.4.3 SAVEPW Command

The SAVEPW command is a TSO CLIST that invokes the DCE client program, SAVEPW, to store a DCE principal's DCE password in his or her associated RACF user profile. The command will only store the currently logged in principal's password in the associated RACF userid. To ensure that the DCE password in the DCE registry and the DCE password in the user's RACF password match, both are updated. The following is the syntax of the command:

```
SAVEPW <sysid> <old password> <new password> <verify password>
```

where:

<sysid> is the name of the MVS system to be updated.

<old password> is the DCE principal's old password

<new password> is the DCE principal's new password

<verify password> is the DCE principal's password re-entered for verification.

Figure 7 illustrates the processing that is done by the SAVEPW command. As is seen from the figure, SAVEPW is implemented as a DCE client/server application. The fragments of code shown are pseudo code. The fragment of the user SVC is another part of the user SVC that was discussed in the previous section.

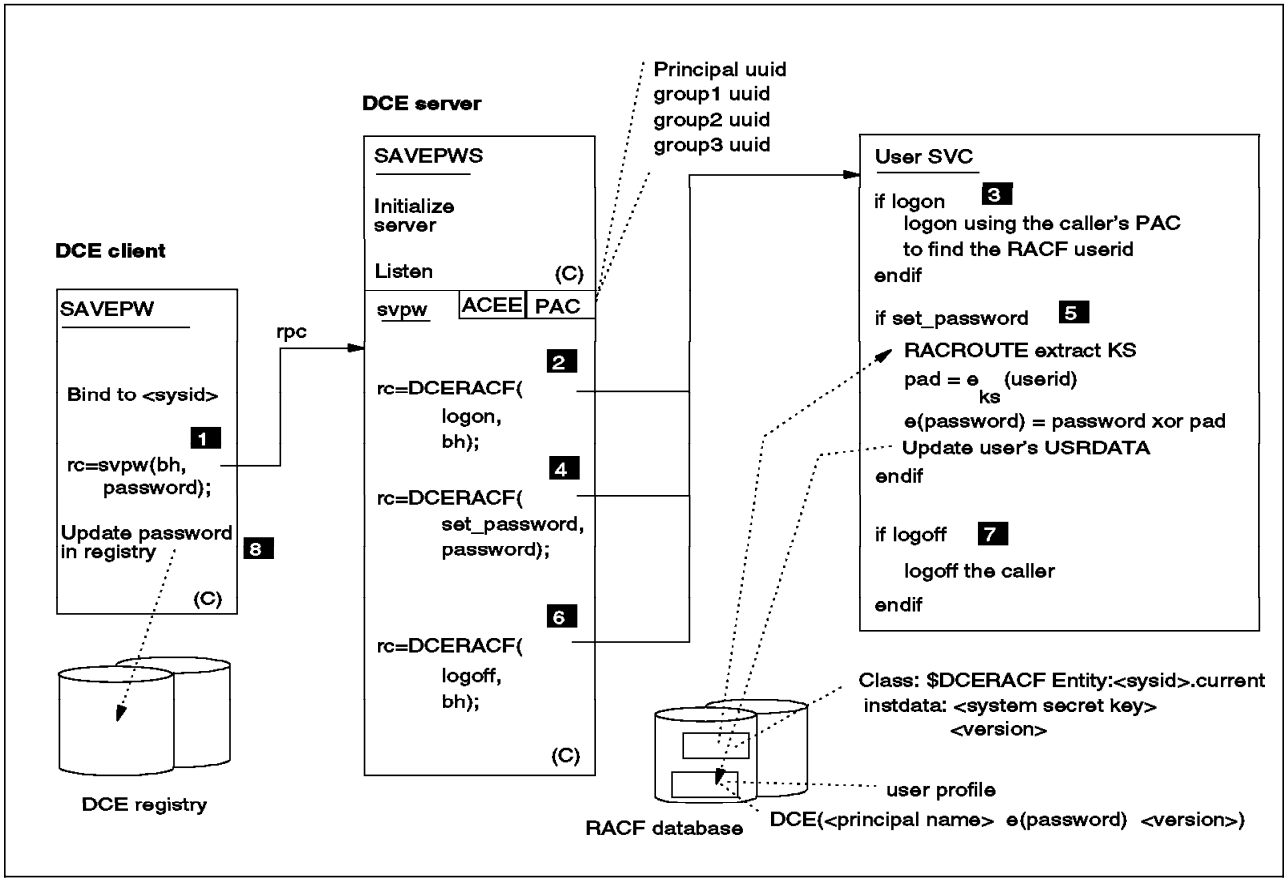


Figure 7. SAVEPW Command

1. SAVEPW is the client portion of the SAVEPWS DCE application server. It has four parameters passed to it by the user, <sysid>, <old password>,

<new password>, and <verification password>. The SAVEPWS application server is replicated on every MVS/RACF system in the DCE cell and provides a RACF user profile updating capability on each RACF system. The server requires the use of authenticated RPCs.

a. If the <new password> and the <verification password> match, the client program must first find the exported bindings for the requested RACF system <sysid>. The exported name in the CDS name space is /.:/dceracf/<sysid>.

b. If the bindings are found, bind to the requested RACF system.

c. The client program then calls the server program svpw (see ″SAVEPW, Call the remote procedure″ on page 173) passing it the following parameters:

```
rc=svpw(bh, password)
```

where:

bh is the explicit binding handle of the request

password is the password to be stored in the client principal's associated RACF user profile.

2. The manager code for svpw makes use of the functions provided by the user SVC (described previously in this book), as an unauthorized MVS APF program. In fact, it provides a working example of an MVS/ESA OpenEdition DCE application server using the facilities of RACF to complement the DCE security.

   The server calls **DCERACF(logon, bh)** to logon the DCE client onto RACF using his or her associated RACF userid (see ″SAVEPWS, Logon client onto RACF″ on page 187).

3. DCERACF calls the user SVC passing it the client's binding handle. The user SVC creates an ACEE for the client's associated RACF userid, as described previously. All subsequent calls to RACF, implied and explicit, will use this ACEE to determine the client's authorization.

4. svpw then calls **DCERACF(RACF_set_password, password)** to set the password in the current RACF user profile.

5. DCERACF calls the user SVC passing it the password. The user SVC processes the set password request as follows:

   a. Retrieve the current system secret encryption key (see ″SETPWD subroutine, extract secret key″ on page 94).

   b. Encrypt the current userid under the system secret encryption key to be used as an encryption pad (see ″SETPWD subroutine, Encrypt userid under secret key″ on page 95).

   c. Build the USRDATA field of the current RACF user. The format of the USRDATA includes the offsets to the DCE principal, the encrypted password, and the version of the system secret encryption key used as the first three bytes of the USRDATA (see ″SETPWD subroutine, Build USRDATA″ on page 95). To hide the DCE password, the encryption pad created in the previous step is exclusive OR'ed with the password.

   d. Update the current RACF user profile, USRDATA (see ″SETPWD subroutine, Update USRDATA″ on page 96).

6. The server calls **DCERACF(logoff, bh)** to logoff the DCE client from RACF.

7. DCERACF calls the user SVC passing it the client's binding handle. The user SVC deletes the current ACEE for the client's associated RACF userid, as described previously (see "LOGOFF subroutine, Issue RACROUTE DELETE" on page 93).
8. On return from the SVPW server function, the SAVEPW client program has to update the DCE registry to ensure that the DCE password that is storedin the RACF database is identical to the DCE password that is stored in the DCE registry (see "SAVEPW, Update DCE registry" on page 175).

1.4.4 Passticket Support

RACF provides support of one-time-use passwords, called passtickets. The passticket server uses that support to generate and return a passticket to a DCE client.

A DCE application server can choose to use the RACF passticket support so that the client is fully authenticated by the RACF passticket, and can be given access to RACF resource independent of the level of access that the DCE application server may have. For example, an application could create a MVS data set and add/update records where the data set name is not known by the application in advance and, therefore, cannot be added to the access list in anticipation of the request.

The passticket support provides:

- A passticket Server, **PTKTS**, that will generate a RACF passticket for the DCE client's associated RACF userid
- A function, **bind_to_passticket_server** in module PTKTLIB, to bind to that server
- A function, **passticket**, to request a RACF passticket as a client to the above server
- An additional option to the RACF_logon function of the user SVC to use the passticket as a RACF one-time-use password

1.4.4.1 Passticket Server

The passticket server, PTKTS (see B.3.4, "PTKTS" on page 142), is a DCE server that will generate a RACF passticket that is passed as a parameter by an application client to its application server. The application server can use the passticket option of the **DCERACF(RACF_logon, bh, passticket)** request to logon the client on to RACF as a fully authenticated RACF user.

This will allow the client to have unrestricted access to all RACF resources to which it has been given access.

This server must be run in an MVS APF authorized library and be APF authorized, so that it can call the privileged RACF service routine that generates RACF passtickets.

The passticket server's manager code, **passticket**, calls a C interface routine, **PTKTGEN**, which builds the DCE application name used by the user SVC, DCE<smf system id> and calls the RACF passticket generator for it. The SMF system id is set by the installation in SID parameter in member SMFPRM00 of SYS1.PARMLIB. Any special characters in that SID are removed. For example, SID=S*Y5 would be used to build DCESY5 application name,

The server advertises its presence in the DCE CDS by an entry of the <sysid> in the /.:/passticket/ directory.

For example:

```
/.:/passticket/<sysid>
```

where the <sysid> is the system name in the MVS CVT field, CVTSNAME. This is set by the installation in SYS1.PARMLIB IEASYS00 member, SYSNAME parameter.

1.4.4.2 Passticket Function Library

The passticket function library, PTKTLIB, contains the **bind_to_passticket_server** and the client request function **passticket**.

The **bind_to_passticket_server** function imports and binds to the requested passticket server on the target MVS system. This function only needs to be issued once during the execution of a client program and returns the binding handle to be used in subsequent calls to this server. Storage for the handle has to allocated in the calling client program. There are multiple handles, one for each different target MVS system.

The **passticket** function requests a one-time-use passticket from the target MVS system. It is called multiple times during the execution of the logic of the application client program. The passticke is passed in the application′s parameter list to its application server. The application server passes it to the user SVC as an authenticator for the client′s logon to RACF.

Chapter 2. Installation

This chapter discusses the installation of the programs associated with the design shown in Chapter 1, “Design” on page 1. The source is distributed on the diskette that accompanies this redbook, and is listed in Appendix B, “Source Code” on page 59. The contents of the diskette is described in appendix A.1, “Contents of the Installation Diskette” on page 53. The MVS data sets that are created are described in appendix A.2, “MVS Data Sets” on page 54.

It is suggested that the steps of the installation follow the order presented here:

- Step 1 - Uploading to the host
  - Step 1a - On OS/2, Execute a:\ALLOCATE.CMD
  - Step 1b - On TSO, Execute hlq.DCERACF.ALLOCATE.CLIST data set
  - Step 1c - On OS/2, Execute a:\UPLOAD.CMD
- Step 2 - Create/merge RACF tables and exit
  - Step 2a - ICHRFR01
  - Step 2b - ICHRRCDE
  - Step 2c - ICHRCX02
- Step 3 - Tailor JCL procedures
  - Step 3a - DCECC
  - Step 3b - DCELK
- Step 4 - Assemble and link user SVC
  - Step 4a - USVCNUM
  - Step 4b - USERSVC
  - Step 4c - Update IEASVC00
  - Step 4d - DCERACF
  - Step 4e - PTKTGEN
- Step 5 - Compile and link utilities
  - Step 5a - DCEKEY
  - Step 5b - DCERACFU
  - Step 5c - SAVEPW
  - Step 5d - SAVEPWS
  - Step 5e - PTKTS
  - Step 5f - PTKTLIB
  - Step 5g - PTSTC
  - Step 5h - PTSTS
- Step 6 - Re-IPL
- Step 7 - Prepare servers on DCE
  - Step 7a - Update DCE Registry
  - Step 7b - Update DCE Cell Directory Server
  - Step 7c - Update DCE Security Server
- Step 8 - Prepare RACF profiles
  - Step 8a - Define administrator profile
  - Step 8b - Define and set system secret key
  - Step 8c - Set up users
  - Step 8d - Define passticket profile
  - Step 8e - Set up passticket server

2.1 Step 1 - Uploading to the Host

The diskette distributed with this redbook contains the source code to be uploaded to the host (see appendix A.1, “Contents of the Installation Diskette” on page 53). This description assumes the following:

- OS/2 on the workstation
- REXX on the workstation
- CM/2 send command is available
- Uploading to TSO

If the above are not available, you will have to upload the files and the ALLOCATE CLIST yourself.

To upload the code:

1. Step 1a - On OS/2, Execute a:\ALLOCATE.CMD

   This command file (see appendix B.1.1, “ALLOCATE.CMD” on page 59) uploads the file a:\ALLOCATE.CLI. It will prompt for the CM/2 host session to be used, and the MVS high level qualifier (hlq) to be used. The data set created on MVS is:

   ```
   hlq.DCERACF.ALLOCATE.CLIST
   ```

2. Step 1b - On TSO, Execute hlq.DCERACF.ALLOCATE.CLIST data set

   This CLIST (see appendix B.1.2, “ALLOCATE.CLIST” on page 59) will allocate all the MVS data sets needed (see appendix A.2, “MVS Data Sets” on page 54). It will prompt for the MVS high level qualifier (hlq), the volume to be used (VOL), and the unit to be used (UNIT).

3. Step 1c - On OS/2, Execute a:\UPLOAD.CMD

   This command file (see appendix B.1.3, “UPLOAD.CMD” on page 60) will upload all the files in the a:\DCERACF directory to the appropriate MVS data set. It will prompt for the CM/2 host session to be used, and the MVS high level qualifier (hlq) to be used.

2.2 Step 2 - RACF Tables

A RACF general resource class, $DCERACF, has to be created for use with the following profiles:

- System secret encryption keys:

  <sysid>.CURRENT, and previous versions

  <sysid>.V001 through <sysid>.Vnnn

- ADMINISTRATOR
- UUID cross reference profiles

Two jobs are provided that can be used (or can be merged into the installation’s RACF table jobs) to create a RACROUTE class definition table and the corresponding RACF installation general resource table.

Note: *These changes will need a re-IPL to come into effect, but that doesn’t need be done until the user SVC has also been assembled and linked into SYS1.LPALIB.*

2.2.1 Step 2a - ICHRFR01

The RACROUTE table, ICHRFR01, indicates the processing that is to be done by MVS SAF. It has to be updated, or created, to add installation's RACF general resource class $DCERACF. Edit job ICHRFR01 (see appendix B.8.9, "ICHRFR01" on page 216) to include it with any other installation classes, modify the JCL to reflect the data set names chosen in step 1b above and submit it.

2.2.2 Step 2b - ICHRRCDE

The RACF user general resource table, ICHRRCDE, defines the $DCERACF class. Edit job ICHRRDCE (see appendix B.8.10, "ICHRRCDE" on page 218) to include it with any other installation classes, modify the JCL to reflect the data set names chosen in step 1b above and submit it.

2.2.3 Step 2c - ICHRCX02

ICHRCX02 is the RACHECK post-processing exit. It is used here to close a security hole introduced by the RACF and DCE Security interoperation code.

Note: If the installation already uses the ICHRCX02 exit, this code will have to be merged with it to ensure the integrity exposure is closed. This code will only process the request if the field ACEEAPLN (the application name) in the ACEE is DCERACF and the job step is non-APF authorized. Therefore, only ACEEs created by the user SVC are affected by this code.

Edit job ICHRCX02 (see appendix B.8.8, "ICHRCX02" on page 215) to include it with any other installation ICHRCX02, modify the JCL to reflect the data set names chosen in step 1b above and submit it.

2.3 Step 3 - Tailor JCL Procedures

DCECC and DCELK JCL procedures (see appendix B.8.1, "DCECC" on page 209 and appendix B.8.4, "DCELK" on page 212) have been tailored to simplify the compilation and linking of DCE programs. Each procedure has symbolic variables defining the naming conventions for the C/370 compiler, LE/370, and DCE.

Edit DCECC and DCELK in hlq.DCERACF.JCL to set those variables appropriately for your installation. These procedures are used by all the following jobs through the use of the statement:

```
//JCLLIB  JCLLIB ORDER=(hlq.DCERACF.JCL)
```

2.4 Step 4 - User SVC

The user SVC, and the programs that use it, have to be assembled and linked into the appropriate libraries.

Note: The assembler programs have been written using a set of structured assembler macros. These macros have been distributed on the diskette for this redbook and copied into the hlq.DCERACF.ASM library during the step 1c above. The macros are in member STRMACS. Member STRREF {see appendix Appendix C, "Structured Programming Macro Reference" on page 231) is a softcopy manual that you may wish to browse or print. It is Bookmaster source, but, it is readable even if you don't have Bookmaster.

2.4.1 Step 4a - USVCNUM

The SVC number chosen by the installation is put into member USVCNUM. That member is copied into all routines that make use of the user SVC when they are assembled.

Edit USVCNUM (see appendix B.2.5, “USVCNUM” on page 80) to set the installation chosen user SVC number and the module name.

The naming convention for user SVC module names is IGC00nns, where *nns* is the signed character SVC number. For example, SVC 255 has a module name of IGC0025E.

2.4.2 Step 4b - USERSVC

Edit and submit job USERSVC (see appendix B.8.24, “USERSVC” on page 230) to assemble and link USERSVC. This job includes the compilation of GETPAC subroutine. The user SVC is linked into SYS1.LPALIB.

2.4.3 Step 4c - Update IEASVC00

Edit SYS1.PARMLIB(IEASVC00) to insert the characteristics of the user SVC using the installation chosen SVC number. For example:

```
SVCPARM 255,REPLACE,TYPE(3),APF(NO)
```

2.4.4 Step 4d - DCERACF

Edit and submit job DCERACF (see appendix B.8.5, “DCERACF” on page 214) to assemble the DCERACF subroutine. The object deck produced will be linked into any C program that needs to use the user SVC.

2.4.5 Step 4e - PTKTGEN

Edit and submit job PTKTGEN (see appendix B.8.11, “PTKTGEN” on page 219) to assemble the PTKTGEN subroutine. The object deck produced will be linked into the passticket server, PTKTS.

2.5 Step 5 - Utilities

2.5.1 Step 5a - DCEKEY

Edit and submit job DCEKEY (see appendix B.8.2, “DCEKEY” on page 211) to assemble and link DCEKEY into SYS1.LINKLIB. The module must be APF authorized because it issues privileged RACF macros.

Every time the program is executed, it generates a new system secret encryption key. Any new DCE password stored in the RACF user profiles will use the new key. Any DCE password already stored is automatically re-encrypted with the new key whenever it is referenced. The previous versions of the key are kept in the RACF database.

The installation may choose to make the program a RACF controlled program to restrict the use of it, but it is not necessary for the system′s security to restrict its use.

2.5.2 Step 5b - DCERACFU

1. Edit and submit job DCERACFU (see appendix B.8.6, “DCERACFU” on page 214) to compile and link the DCERACFU utility. The link steps of this job shows the use of the DCERACF object deck.
2. Edit and copy DCERACFU CLIST (see appendix B.7.1, “DCERACFU” on page 207) into an installation CLIST library available to all TSO users.

2.5.3 Step 5c - SAVEPW

1. Edit and submit job SAVEPW (see appendix B.8.20, “SAVEPW” on page 227) to compile and link SAVEPW client program. This job also compiles the client stub (SVPWCS) of the SVPW application DCE interface.

   This client program is written to be portable to other platforms. The files SAVEPW.C, SVPWCS.C, and SVPW.H on the diskette can be copied onto a UNIX or OS/2 system and compiled and linked with DCE for that platform.
2. Edit and copy SAVEPW CLIST (see appendix B.7.2, “SAVEPW” on page 207) into an installation CLIST library available to all TSO users.

2.5.4 Step 5d - SAVEPWS

Edit and submit job SAVEPWS (see appendix B.8.21, “SAVEPWS” on page 228) to compile and link SAVEPWS server program. This job also compiles the server stub (SVPWSS) of the SVPW application DCE interface.

This server program can only execute on an MVS/ESA OpenEdition DCE platform with the code provided by this redbook.

2.5.5 Step 5e - PTKTS

Edit and submit job PTKTS (see appendix B.8.14, “PTKTS” on page 221) to compile and link PTKTS server program. This job also compiles the server stub, PTKTSS, of the **PTKT** application DCE interface.

This server program can only execute on an MVS/ESA OpenEdition DCE platform with the code provided by this redbook.

2.5.6 Step 5f - PTKTLIB

Edit and submit job PTKTLIB (see appendix B.8.13, “PTKTLIB” on page 220) to compile the PTKTLIB functions for use by a client program that wants to use the passticket server, PTKTS.

2.5.7 Step 5g - PTSTC

Edit and submit job PTSTC (see appendix B.8.16, “PTSTC” on page 223) to compile and link the PTSTC test and demonstration client program that uses the passticket server, PTKTS.

2.5.8 Step 5h - PTSTS

Edit and submit job PTSTS (see appendix B.8.18, “PTSTS” on page 225) to compile and link the PTSTS test and demonstration server program that uses the passticket server, PTKTS.

2.5.9 Step 6 - Re-IPL

Re-IPL the MVS system at a convenient time to bring the changes made into effect. The programs in this installation process cannot be executed until the system has been re-IPLed.

2.6 Step 7 - Prepare Servers on DCE

The SAVEPW and PTSTC client and SAVEPWS, PTKTS, and PTSTS server programs have been written assuming the following:

- The servers SAVEPWS, PTKTS, and PTSTS, execute a DCE login for DCE principals savepw, passticket, and ptst respectively.
- SAVEPWS will export its bindings to **/.:/dceracf/<sysid>**.
- PTKTS will export its bindings to **/.:/passticket/<sysid>**.
- PTSTS will export its bindings to **/.:/passticket/ptst/<sysid>**.

where <sysid> is the MVS system name in SYS1.PARMLIB(IEASYS00) parameter SYSNAME.

To create the environment for a server, the DCE utilities, **rgy_edit**, **cdscp**, and **acl_edit** are used.

2.6.1 Step 7a - DCE Registry

Use *rgy_edit* to:

1. Create DCE principal and account, *savepw*.
2. Create DCE principal and account, *passticket*.
3. Create DCE principal and account, *ptst*.
4. Create Keytab file to contain the encrypted DCE password for the SAVEPWS server.
5. Create Keytab file to contain the encrypted DCE password for the PTKTS server.
6. Create Keytab file to contain the encrypted DCE password for the PTSTS server.
7. Make *savepw* principal a member of *subsys/dce/rpc-server-group*
8. Make *passticket* principal a member of *subsys/dce/rpc-server-group*
9. Make *ptst* principal a member of *subsys/dce/rpc-server-group*

2.6.2 Step 7b - DCE Cell Directory

Use *cdscp* to:

1. Create a directory */.:/dceracf*
2. Create a directory */.:/passticket*
3. Create a directory */.:/passticket/ptst*
4. Create an object */.:/dceracf/<sysid>*, where <sysid> is the MVS system name (see SYS1.PARMLIB(IEASYS00), SYSNAME).
5. Create an object */.:/passticket/<sysid>*, where <sysid> is the MVS system name (see SYS1.PARMLIB(IEASYS00), SYSNAME).

6. Create an object */.:/passticket/ptst/<sysid>*, where <sysid> is the MVS system name (see SYS1.PARMLIB(IEASYS00), SYSNAME).

2.6.3 Step 7c - DCE Security Server

Use *acl_edit* to:

1. Give principal *savepw*, *rwdtci* authority for */.:/dceracf*
2. Give principal *passticket*, *rwdtci* authority for */.:/passticket*
3. Give principal *ptst*, *rwdtci* authority for */.:/passticket/ptst*
4. Give principal *savepw*, *rwdtc* authority for */.:/dceracf/<sysid>*
5. Give principal *passticket*, *rwdtc* authority for */.:/passticket/<sysid>*
6. Give principal *ptst*, *rwdtc* authority for */.:/passticket/ptst/<sysid>*
7. Copy a DCE ticket cache HFS file for batch execution of SAVEPWS.
8. Copy a DCE ticket cache HFS file for batch execution of PTKTS.
9. Copy a DCE ticket cache HFS file for batch execution of PTSTS.

2.7 Step 8 - Prepare RACF Profiles

The following steps will create the set of RACF profiles that control the operation of the RACF and DCE security interoperation.

2.7.1 Step 8a - Define Administrator profile

The ADMINISTRATOR profile controls the use of several utilities. To define to RACF, issue the following commands (we assume that the user has RACF system special authority):

```
SETROPTS CLASSACT($DCERACF)
RDEFINE $DCERACF ADMINISTRATOR UACC(NONE)
PERMIT ADMINISTRATOR CLASS($DCERACF) ACC(ALTER) ID(.....)
```

2.7.2 Step 8b - Define and Set System Secret Key

The system secret key is used to encrypt the DCE password stored in the user's RACF profile USRDATA field. A pool of system secret key profiles are defined as follows:

```
RDEFINE $DCERACF sysid.CURRENT UACC(NONE)
RDEFINE $DCERACF sysid.V001    UACC(NONE)
RDEFINE $DCERACF sysid.V002    UACC(NONE)
RDEFINE $DCERACF sysid.V003    UACC(NONE)
RDEFINE $DCERACF sysid.V004    UACC(NONE)
RDEFINE $DCERACF sysid.V005    UACC(NONE)
```

where:

sysid is SYSNAME parameter in IEASYS00 member of SYS1.PARMLIB data set.

Once these have been defined, edit and submit job DCEKEYR (see appendix B.8.3, "DCEKEYR" on page 211).

2.7.3 Step 8c - Set Up Users

The procedure is divided into three parts, setting up CLISTs, administrator actions and enduser actions.

1. There are two CLISTs provided, DCERACFU and SAVEPW. These CLISTs have t be copied into a general use CLIST library, or the hlq.DCERACF.CLIST has to be added to the concatenation of SYSPROC data sets in the user′s TSO logon procedure, or the users will have to add it their own concatenation of SYSPROC data sets.
2. The administrator logs in to DCE as either cell_admin or DCE principal with the same authority as cell_admin to alter DCE principals. He or she issues the following commands for each user to be defined:

   ```
   DCERACFU set racf <DCE principal> <RACF userid>
   DCERACFU set dce  <DCE principal> <RACF userid>
   ```

   This will create the USRDATA field in the user′s RACF user profile and create a cross reference UUID profile.
3. The user then can enter his or her DCE password for the first time as follows:

   ```
   DCERACFU set password <DCE password> <verify password>
   ```

   On subsequent changes to the DCE password, he or she can use the SAVEPW command from any system in the DCE cell.

   The user may choose to automatically login to DCE when ever he or she logs onto TSO by inserting the following command on the TSO logon panel command field:

   ```
   DCERACFU DCELOGIN
   ```

   This will use the stored DCE password from the user′s RACF user profile to login his or her associated DCE principal onto DCE.

2.7.4 Step 8d - Define Passticket Profile

The passticket server has to have a RACF profile defined for the application name that representing DCE entry to the MVS system. The passtickets combine the user′s RACF userid and the name of this application name to generate a one-time-use password. The application name used by the user SVC is DCE<smfid>, where <smfid> is the SID parameter in SMFPRM00 member of SYS1.PARMLIB. The <smfid> must have blanks and special characters removed. The following commands will define the encryption key represented by the application name:

```
SETROPTS CLASSACT(PTKTDATA)
RDEFINE PTKTDATA DCE<smfid> SSIGNON(KEYMASKED(XXXXXXXXXXXXXXXX))
SETROPTS RACLIST(PTKTDATA) REFRESH
```

Where:

xxx...xxx is 16 hexadecimal digits that is used as the encryption key.

2.7.5 Step 8e - Set up Passticket Server

The passticket server will need to be started when DCE is started on the MVS system. It can be set up as an MVS started task, or submitted by the system operation group.

Edit PTKTSRUN (see appendix B.8.15, “PTKTSRUN” on page 222) as appropriate to your installation.

Chapter 3. Using DCERACF Subroutine and the Utilities

This chapter discusses the use of:

- the DCERACF subroutine

and the utilities:

- DCEKEY, which generates a new system secret encryption key
- DCERACFU, which associates DCE principals with RACF userids
- SAVEPW, which enables an enduser to change his or her DCE password.
- PTKTS, and PTKTLIB which enables an application client to request a RACF passticket for use with its application server.

3.1 Using DCERACF Subroutine

The DCERACF subroutine is an interface routine for 'C' programs to the user SVC.

The user SVC provides access to authorized RACF services as a part of the RACF/DCE interoperation. The functions it provides are described below.

Note: Member DCERACF in hlq.DCERACF.H contains C defines for the REQUEST values for the user SVC functions as follows:

```
#define RACF_logon                  1
#define RACF_check_authorisation    2
#define RACF_check_authorization    2
#define RACF_logoff                 3
#define RACF_set_DCE_password       4
#define RACF_get_DCE_password       5
#define RACF_get_DCE_principal      6
#define RACF_set_DCE_principal      7
#define RACF_get_DCE_usrdata        8
#define RACF_get_MVS_sysid          9
#define RACF_set_DCE_uuid          10
#define RACF_get_DCE_uuid          11
```

The header file contains functions that will convert a return code into the appropriate message for the requested function. These functions only provide the message string. The application can use the returned message in a printf function or pass it back to the client in a parameter.

The header file also contains a sample function, *dcelogin* that uses the principal and DCE password to login the RACF user into DCE.

3.1.1 Logon to RACF

Logon the client onto RACF. The client's UUID is extracted from the PAC (privilege attribute certificate) in the binding handle. The UUID is converted into an UUID_string to find the UUID cross reference profile, from which the RACF userid is extracted.

The application must ensure that DCERACF(RACF_logoff,...) is called before this client's processing is completed.

```
rc = DCERACF(RACF_logon,
             bh,
             ticket)
```

Parameters:

RACF_logon is the function being requested

bh binding handle (opaque).

The binding handle passed by the DCE application server enables the GETPAC subroutine to communicate with the DCE security server to find the RACF userid associated with this principal.

ticket RACF one-time-use passticket (optional)

rc is the return code as follows:

0 RACF logon OK, ACEE created

4 RACF userid has been revoked

8 Invalid password or RACF userid not known to RACF

12 DCE client is not using an authenticated RPC

16 DCE client does not have a RACF UUID cross reference profile

20 Requested function not recognized

The message mapping function is *RACF_logon_error(rc, msg)*

3.1.2 Check Authorization to a Resource

Explicitly check whether this client is allowed to access a RACF protected resource in the nominated RACF class at the requested level.

For a non-APF authorized server, both the client's ACEE and the application server's ACEE are checked to ensure that a potential integrity exposure introduced by the user SVC is closed. The server must have at least the level of access as the client.

Notes:

1. An APF authorized server has no restrictions placed on it.
2. A fully authenticated RACF user that has used a RACF passticket has no restrictions placed on it.

```
rc = DCERACF(RACF_check_authorization,
             bh,
             class,
             entity,
             access)
```

Parameters:

RACF_check_authorization is the function being requested

bh binding handle (opaque)

class class of the resource (null terminated string)

entity resource to be tested (null terminated string)

access access requested (null terminated string)

READ

WRITE

CONTROL

ALTER

rc is the return code as follows:

0 access granted

8 access denied

12 resource is not known to RACF

16 class is not known to RACF

20 Requested function not recognized

The message mapping function is *RACF_check_authorization_error(rc, msg)*.

3.1.3 Logoff RACF

Logoff the client from RACF. Only the current TCB ACEE is able to be logged off.

```
rc = DCERACF(RACF_logoff,
             bh)
```

Parameters:

RACF_logoff is the function being requested

bh binding handle (opaque)

rc is the return code as follows:

0 logged off RACF

8 userid not logged on

20 Requested function not recognized

The message mapping function is *RACF_logoff_error(rc, msg)*

3.1.4 Set Encrypted DCE Password

Set the encrypted DCE password in the current logged on RACF userid, user profile, and USRDATA field. If there is no TCB ACEE, the address space ACEE is used.

Note: The length of the principal name plus the length of the encrypted DCE password cannot be greater 248 in this implementation, even though DCE allows a maximum of 1023 for each.

```
rc = DCERACF(RACF_set_DCE_password,
             password)
```

Parameters:

RACF_set_DCE_password is the function being requested

password clear text to be encrypted and written into the user's RACF profile and ACEE (null terminated string) (input)

rc is the return code as follows:

0 Normal

8 RACF user profile USRDATA field does not contain DCE principal name, DCE password, and key version fields or the DCE fields are not valid.

12 RACF <sysid>.CURRENT profile not found

16 Unable to encrypt pad

20 Requested function not recognized

28

Unable to update user profile

The message mapping function is *RACF_set_DCE_password_error(rc, msg)*.

3.1.5 Get DCE Password

Get and decrypt the encrypted DCE password from the currently logged on RACF userid, user profile, USRDATA field. If there is no TCB ACEE, the address space ACEE is used.

```
rc = DCERACF(RACF_get_DCE_password,
             password)
```

Parameters:

RACF_get_DCE_password is the function being requested

password decrypted DCE password (null terminated string) (output)

rc is the return code as follows:

0 Normal

8 RACF user profile USRDATA field does not contain DCE principal name, DCE password, and key version fields or the DCE fields are not valid.

12 RACF ′<sysid>.Vnnn′ profile not found

16 Unable to encrypt pad

20 Requested function not recognized

The message mapping function is *RACF_get_DCE_password_error(rc, msg)*.

3.1.6 Get DCE Principal

Get the DCE principal from the currently logged on RACF userid, user profile, and USRDATA field. If there is no TCB ACEE, the address space ACEE is used.

```
rc = DCERACF(RACF_get_DCE_principal,
             principal)
```

Parameters:

RACF_get_DCE_principal is the function being requested from the current ACEE

principal DCE principal name (null terminated string) (output)

rc is the return code as follows:

0 Normal

8 DCE keyword not found in USRDATA

12 DCE field in USRDATA is invalid

20 Requested function not recognized

The message mapping function is *RACF_get_DCE_principal_error(rc, msg)*.

3.1.7 Set DCE Principal

Set the DCE principal name in the nominated RACF userid, user profile, and USRDATA field.

Note: The length of the principal name plus the length of the encrypted DCE password cannot be greater 248 in this implementation, even though DCE allows a maximum of 1023 for each.

```
rc = DCERACF(RACF_set_DCE_principal,
             userid,
             principal)
```

Parameters:

RACF_set_DCE_principal is the function being requested

userid RACF userid (null terminated string) (input)

principal DCE principal name (null terminated string) (input)

rc is the return code as follows:

0 Normal

8 DCE keyword not found in USRDATA

12 DCE field in USRDATA is invalid

16 Caller is not authorized to this request

20 Requested function not recognized

The message mapping function is *RACF_set_DCE_principal_error(rc, msg)*.

Note: The caller must have ALTER access to the ADMINISTRATOR profile in class $DCERACF.

3.1.8 Get USRDATA

Get the USRDATA field from the nominated RACF userid, user profile. The USRDATA field is returned in other character or hex format.

Note: The caller must have ALTER access to the ADMINISTRATOR profile in class $DCERACF.

```
rc = DCERACF(RACF_get_DCE_usrdata,
             userid,
             format,
             area)
```

Parameters:

RACF_get_DCE_usrdata is the function being requested

userid RACF userid (null terminated string) (input)

format Format of the output. C for character, anything else for hex.

area contents of USRDATA field of the requested userid (null terminated string) (output)

rc is the return code as follows:

0 Normal

16 Caller is not authorized to this request

20 Requested function not recognized

The message mapping function is *RACF_get_DCE_usrdata_error(rc, msg)*.

3.1.9 Get MVS Sysid

Get the MVS system ID from the MVS CVT, CVTSNAME field. This field is set by the installation in member IEASYS00 in SYS1.PARMLIB parameter, SYSNAME. It is used by an application server in building an exported binding name in the DCE CDS.

```
rc = DCERACF(RACF_get_MVS_sysid,
             sysid)
```

Parameters:

RACF_get_MVS_sysid is the function being requested

sysid System ID from MVS CVT (null terminated string) (output)

rc is the return code as follows:

0 Normal

20 Requested function not recognized

The message mapping function is *RACF_get_MVS_sysid_error(rc, msg)*.

3.1.10 Set DCE UUID

Create, or overwrite, a UUID cross reference profile associating a RACF userid with a DCE principal. The RACF userid is stored in the INSTDATA field of the profile.

Note: The caller must have ALTER access to the ADMINISTRATOR profile in class $DCERACF.

```
rc = DCERACF(RACF_set_DCE_UUID,
             UUID,
             userid)
```

Parameters:

RACF_set_DCE_UUID is the function being requested

UUID UUID string (input) (null terminated)

userid associated RACF userid (input) (null terminated)

rc is the return code as follows:

0 Normal

8 Unable to create/replace UUID profile

16 Not authorized

20 Requested function not recognized

The message mapping function is *RACF_set_DCE_UUID_error(rc, msg)*.

3.1.11 Get DCE UUID

Get the RACF userid from the nominated UUID cross reference profile. The RACF userid is stored in the INSTDATA field of the profile.

```
rc = DCERACF(RACF_get_DCE_UUID,
             UUID,
             userid)
```

Parameters:

RACF_get_DCE_UUID is the function being requested

UUID UUID string (input) (null terminated)

userid associated RACF userid (output) (null terminated)

rc is the return code as follows:

0 Normal

8 Unable to find UUID profile

16 not authorized

20 Requested function not recognized

The message mapping function is *RACF_set_DCE_UUID_error(rc, msg)*.

3.2 Using the DCEKEY Utility

DCEKEY generates a system secret encryption key that is used to encrypt a user's DCE password. The system secret key is stored in the RACF database, in a RACF general resource profile in the $DCERACF class. The encryption key is stored in INSTDATA field of the profile. The naming conventions used for the profiles are:

<sysid>.CURRENT

or

<sysid>.Vnnn

The utility assumes the following:

- The RACF administrator has created a RACF class of $DCERACF.
- The RACF administrator has created a pool of general resource profiles in that class, conforming to the naming convention above. The minimum number of profiles is two: one for <sysid>.CURRENT and one for the version number copy of that profile. The version number copy profile is used to save previous keys and enable the decryption of the old DCE password with re-encryption with the current key when a user next logs into DCE.
- There should be at least one more version number profile than the number of times that this utility has been executed.

Every time this utility executes, it will create a new system secret encryption key. Any new DCE passwords stored in the RACF data base will use the <sysid>.CURRENT key. Whenever a password is retrieved and the key version is different than <sysid>.CURRENT, the password is re-encrypted under the current key.

The program is normally run as a batch job by the RACF administrator. Sample JCL follows:

```
//ERICFIN1 JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=A,REGION=5000K,
//    MSGLEVEL=(1,1)
//*
//* CREATE A NEW SECRET SYSTEM ENCRYPTION KEY
//* ----------------------------------------
//*
//DCEKEYR  EXEC PGM=DCEKEY
//SYSUDUMP  DD  SYSOUT=*
```

3.3 Using the DCERACFU Utility

The DCERACFU utility has several functions:

For an enduser:

1. It allows an end user to set his or her DCE password (encrypted) in his or her RACF profile USRDATA field. To ensure that the password known to the DCE security server is the same, the password is also updated in the DCE registry. If the password has never been set, only the RACF USRDATA field is updated.
2. It allows an end user to login to DCE using the DCE principal name and the encrypted DCE password in his RACF user profile USRDATA. It is used in a TSO CLIST executed automatically when a user logon to TSO and thus giving an effective single network logon.

For an administrator, who must have ALTER access to a RACF general resource ADMINISTRATOR in class $DCERACF:

3. It allows an administrator to control who can implicitly login to DCE. The administrator uses this utility to set the associated DCE principal name in the USRDATA field of the nominated RACF user profile.
4. It allows an administrator to control who can implicitly logon to RACF. The administrator uses this utility to create a ″cross reference UUID″ profile and set the RACF userid associated with a DCE principal
5. It allows an administrator to list the DCE segment stored in the USRDATA field of the nominated RACF userid.
6. It allows an administrator to list the “cross reference UUID” profile for a DCE principal.

The parameters to DCERACFU are subcommands that are passed on the command line of the DCERACFU CLIST as follows:

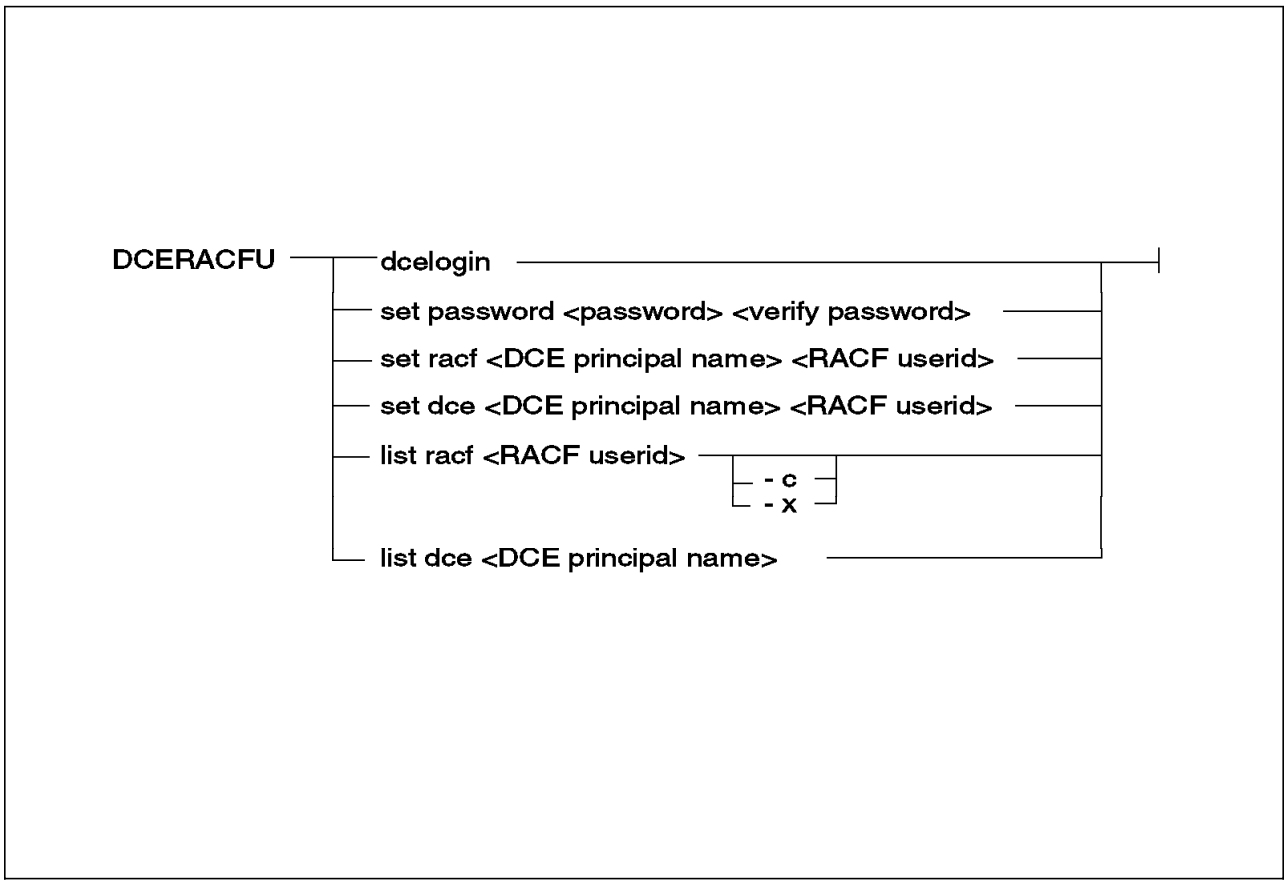


Figure 8. DCERACFU Utility Syntax

The subcommands may also be used in the PARM field in JCL. For example:

```
//jobname JOB ......
// EXEC PGM=DCERACFU,PARM='dcelogin'
//STEPLIB DD DSN=hlq.DCERACF.LOAD,DISP=SHR
```

Note: DCE principal name and password are case sensitive. When editing JCL with the ISPF editor, ensure that CAPS OFF is set.

3.3.1 DCE Login Using DCE Principal and Password in RACF User Profile

The dcelogin subcommand executes a DCE login for the current RACF user, TSO or batch, using the DCE principal name and encrypted DCE password from the RACF user profile, USRDATA field. The user is not prompted for any parameters. It is intended to be used in an automatically executed CLIST from the TSO LOGON panel.

```
DCERACFU dcelogin
```

There are no parameters.

3.3.2 Setting an Encrypted DCE Password

DCERACFU has to be executed under the target RACF userid TSO session only. It is an alternative to the use of the SAVEPW client program. If the password field has already been set, it is assumed that the user is changing his or her password. The old password in the RACF user profile is used to login to DCE so that the password is changed in both places.

```
DCERACFU set password <new password> <verify password>
```

Note: The following parameters are positional:

Parameters:

set specifies the command

password specifies the password subcommand

new password the new password to be set in the current RACF user profile

verify password the new password re-entered to verify the typing

3.3.3 Setting a DCE Principal Name in the RACF User Profile

This subcommand sets the DCE principal name in the RACF user's profile, USRDATA. This creates the USRDATA field without a DCE password as indicated by a zero offset to the password and encryption key version number. Running it against a RACF user profile that has had the DCE password set by the user will lose that password and the user will have to set the password again. This provides a simple means to reset a password when a user has made a mistake.

```
DCERACFU set racf <DCE principal name> <RACF userid>
```

Note: The following parameters are positional:

Parameters:

set specifies the command

racf specifies the racf subcommand

DCE principal name is the user's DCE principal name. It is up to 128 bytes long.

RACF userid is the RACF userid to be associated with the DCE principal name

3.3.4 Setting a RACF Userid in the UUID Cross Reference Profile

This subcommand sets the RACF userid in the UUID cross reference profile, INSTDATA field. The subcommand converts the DCE principal name into a UUID string and creates or overwrites the UUID cross reference profile.

```
DCERACFU set dce  <DCE principal name> <RACF userid>
```

Note: The following parameters are positional

Parameters:

set specifies the command

dce specifies the dce subcommand

DCE principal name is the user's DCE principal name. It is up to 128 bytes long.

RACF userid is the RACF userid to be associated with the DCE principal name

3.3.5 Listing DCE Details in the RACF User Profile

This subcommand enables the administrator to list the contents of the nominated RACF user profile, USRDATA field in character or hex form.

```
DCERACFU list racf <RACF userid> <-c|-x>
```

Note: The following parameters are positional

Parameters:

list specifies the command

racf specifies the racf subcommand

RACF userid is the RACF userid to be listed

-c|-x specifies the form of the listing, character or hex.

3.3.6 Listing RACF Details in the UUID Cross Reference Profile

This subcommand enables the administrator to list the contents of the nominated UUID cross reference profile, INSTDATA field. The subcommand converts the DCE principal name into an UUID string to get the associated RACF userid.

```
DCERACFU list dce  <DCE principal name>
```

Note: The following parameters are positional

Parameters:

list specifies the command

dce specifies the dce subcommand

DCE principal name is the DCE principal name used to find the UUID cross reference profile to be listed

3.4 Using the SAVEPW Utility

The SAVEPW command is a TSO CLIST that invokes the DCE client program, SAVEPW, to store a DCE principal's DCE password in his or her associated RACF user profile. The command will only store the currently logged in principal's password in the associated RACF userid. To ensure that the DCE password in the DCE registry and the DCE password in the user's RACF password match, both are updated.

```
SAVEPW <sysid> <old password> <new password> <verify password>
```

Note: The following parameters are positional:

Parameters:

sysid is the name of the MVS system to be updated.

old password is the DCE principal's old password

new password is the DCE principal's new password

verify password is the DCE principal's password re-entered for verification

3.4.1 Starting the SAVEPWS Server

The SAVEPWS server must be executing to enable the SAVEPW client to process the user's request.

The program is normally run as a batch job. Sample JCL follows:

```
//ERICFINS JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=64M,
//    MSGLEVEL=(1,1)
//* ------------------------------------------------------------------*
//*                                                                   *
//* Note: CHANGE 'hlq.DCERACF' TO '???.DCERACF' ALL                   *
//*                                                                   *
//* Note: The PARM is the HFS KEYTAB file that has the 'savepw'       *
//*       principal entry.                                            *
//*                                                                   *
//* Note: EUVSJRB5 DD points to the ticket cache to be used.          *
//*                                                                   *
//* ------------------------------------------------------------------*
//*                                                                   *
//* EXECUTE THE SAVEPW SERVER                                         *
//* -------------------------                                         *
//*                                                                   *
//SAVEPWS  EXEC PGM=SAVEPWS,REGION=64M,
//             PARM='//u/ericfin/savepw.key'
//STEPLIB  DD DSN=hlq.DCERACF.LOAD,DISP=SHR
//EUVSKRB5 DD PATH='/u/ericfin/krb5ccname.savepw'
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//*
```

3.4.2 Stopping the SAVEPWS Server

An administrator can use the SAVEPW command to stop the SAVEPWS server. The server will check that the client has ALTER authority to RACF general resource, ADMINISTRATOR in class $DCERACF

```
SAVEPW <sysid> stop
```

Note: The following parameters are positional

Parameters:

sysid is the name of the MVS system

stop requests that the SAVEPWS server be stopped

3.5 Using the Passticket Server

RACF provides support of one-time-use passwords, called passtickets. The passticket server uses that support to generate and return a passticket to a DCE client.

A DCE application server can choose to use the RACF passticket support so that the client is fully authenticated by the RACF passticket, and can be given access

to RACF resources independent of the level of access that the DCE application server may have.

The passticket support provides:

- A passticket server, **PTKTS**, that will generate a RACF passticket for the DCE client's associated RACF userid
- A function, **bind_to_passticket_server** in module PTKTLIB, to bind to that server
- A function, **passticket**, to request a RACF Passticket as a client to the above server

A sample application that uses the passticket support is provided (see B.3.5, "PTSTC" on page 153 and B.3.6, "PTSTS" on page 158). The application can create an MVS data set and add records to the end of the data set. The data set name is not known by the application in advance and, therefore, the application server's RACF userid cannot be added to the access list in anticipation of the request; however, the client can create and use the data set because, through the passticket, he or she is a fully authenticated RACF user.

To run the test application:

1. Edit and submit PTSTSRUN job (see B.8.19, "PTSTSRUN" on page 226).
2. Edit and submit PTKTSRUN job (see B.8.15, "PTKTSRUN" on page 222).
3. Issue TSO command:

   ```
   CALL 'hlq.DCERACF.LOAD(PTSTC)' '<sysid> <dataset name> "<record>"'
   ```

   where

   <sysid> is the target MVS system name (SYSNAME in IEASYS00 of SYS1.PARMLIB)

   <dataset name> is the fully qualified MVS data set name, without quotes, to which the record is to be added. If the data set is preallocated, it must be RECFM=VB, LRECL=255, BLKSIZE=2550.

   <record> is the record to be added to the data set. It must be enclosed with double quotes, so that C will parse it as one parameter.

   To stop the PTSTS server, an administrator (ALTER access to ADMINISTRATOR general resource in class $DCERACF) can issue the above CALL command with a data set name of STOP.

3.5.1 Binding to a Passticket Server

The **bind_to_passticket_server** function imports and binds to the requested passticket server on the target MVS system. This function only needs to be issued once during the execution of a client program and returns the binding handle to be used in subsequent calls to this server. Storage for the handle is allocated in the calling client program. There are multiple handles, one for each different target MVS system.

```
#include "ptktlib.h"
handle_t  PTKT_bh;

     bind_to_passticket_server(
        &PTKT_bh,
        sysid);
```

Note: ptktlib.h includes ptkt.h.

Parameters:

PTKT_bh is the binding handle for this passticket server

sysid is the MVS system identity (SYSNAME parameter in member IEASYS00 of SYS1.PARMLIB) of the target MVS system.

3.5.2 Requesting a Passticket

The **passticket** function requests a one-time-use passticket from the target MVS system. It is called multiple times during the execution of the logic of the application client program. The passticket is passed with the application's parameter list to the application server. The application server passes it to the user SVC as an authenticator for the client's logon to RACF.

```
#include "ptktlib.h"
handle_t  PTKT_bh;
char      ticket[9];

     rc = passticket(
             PTKT_bh,
             ticket);
```

Note: ptktlib.h includes ptkt.h.

Parameters:

PTKT_bh is the binding handle for this passticket server

ticket is the RACF passticket (output)

rc is the return code. A nonzero return indicates failure to generate a ticket.

3.5.3 Starting the Passticket Server

The passticket server, PTKTS, must be executing to enable a passticket client to process the user's request.

The program is normally run as a batch job. Sample JCL is a follows:

```
//ERICFINP JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=64M,TIME=1440,
//    MSGLEVEL=(1,1)
//* ------------------------------------------------------------------*
//*                                                                   *
//* Note: The PARM is the HFS KEYTAB file that has the 'passticket'   *
//*       principal entry.                                            *
//*                                                                   *
//* Note: EUVSJRB5 DD points to the ticket cache to be used.          *
//*                                                                   *
//* ------------------------------------------------------------------*
//*                                                                   *
//* EXECUTE THE PTKT SERVER                                           *
//* -----------------------                                           *
//*                                                                   *
//PTKTS EXEC PGM=PTKTS,REGION=64M,
//             PARM='//u/ericfin/passticket.key'
//EUVSKRB5 DD PATH='/u/ericfin/krb5ccname.passticket'
//SYSPRINT DD SYSOUT=*
//*
```

3.5.4 Stopping the Passticket Server

The passticket server does not have any function call to shutdown, The program is stopped by cancelling it.

Appendix A. File and Data Set descriptions

A.1 Contents of the Installation Diskette

The following are directory listings of the installation diskette.

```
 Volume in drive A has no label.

 Directory of A:\

DCERACF      <DIR>      3-03-95  11:52
ALLOCATE CLI     1154  29-01-95  12:49
ALLOCATE CMD      365  29-01-95   9:32
README   TXT     4599   7-02-95  10:55
UPLOAD   CMD     1183  29-01-95  14:22
         5 file(s)       7301 bytes used

 Directory of A:\DCERACF

.            <DIR>      3-03-95  11:52
..           <DIR>      3-03-95  11:52
DCECC    JCL     6068   9-03-95   9:07
DCEKEY   ASM    40836   3-03-95   9:52
DCEKEY   JCL      821  10-03-95  13:25
DCEKEYR  JCL      738   3-03-95  10:25
DCELK    JCL     6150   9-03-95   9:07
DCERACF  ASM    10250   3-03-95   9:52
DCERACF  H      17306   9-03-95  10:16
DCERACF  JCL      668  10-03-95  13:26
DCERACFU C      28907   9-03-95  10:17
DCERACFU CLI      410   3-03-95   9:54
DCERACFU JCL     1005  10-03-95  13:27
GETPAC   C       8533   9-03-95  10:17
GETPAC   JCL     1105  10-03-95  13:29
ICHRCX02 ASM    20418   3-03-95   9:52
ICHRCX02 JCL     1805  10-03-95  13:29
ICHRFR01 JCL     7298   3-03-95   9:55
ICHRRCDE JCL     8282   3-03-95   9:55
PTKT     ACF       44   9-03-95  10:16
PTKT     H        781   9-03-95  10:16
PTKT     IDL      283   9-03-95  10:18
PTKTCS   C       4953   9-03-95  10:18
PTKTGEN  ASM    19762   3-03-95   9:52
PTKTGEN  JCL      710  10-03-95  13:31
PTKTIDL  JCL      738  10-03-95  13:32
PTKTLIB  C       9250   9-03-95  10:16
PTKTLIB  H        307   9-03-95  10:16
PTKTLIB  JCL     1107  10-03-95  13:32
PTKTS    C      23205   9-03-95  10:17
PTKTS    JCL     1495  10-03-95  13:33
PTKTSRUN JCL     1315  10-03-95  13:33
PTKTSS   C       5004   9-03-95  10:19
PTST     ACF       44   9-03-95  10:16
PTST     H        944   9-03-95  10:16
PTST     IDL      368   9-03-95  10:18
```

```
PTSTC    C      12044   9-03-95  10:17
PTSTC    JCL     2267  10-03-95  13:33
PTSTCRUN JCL      425  10-03-95  13:34
PTSTCS   C       7551   9-03-95  10:18
PTSTIDL  JCL      738  10-03-95  13:35
PTSTS    C      24433   9-03-95  10:17
PTSTS    JCL     1412  10-03-95  13:35
PTSTSRUN JCL     1373  10-03-95  13:35
PTSTSS   C       9778   9-03-95  10:17
PTSTSTOP JCL      396  10-03-95  13:36
SAVEPW   C      17620   9-03-95  10:17
SAVEPW   CLI      410   3-03-95   9:54
SAVEPW   JCL     1435  10-03-95  13:36
SAVEPWS  C      24331   9-03-95  10:17
SAVEPWS  JCL     1426  10-03-95  13:36
SAVEPWSR JCL     1381  10-03-95  13:37
STRMACS  ASM    76014   3-03-95   9:52
STRREF   ASM    87166   3-03-95   9:52
SVPW     ACF       44   9-03-95  10:16
SVPW     H        771   9-03-95  10:16
SVPW     IDL      272   9-03-95  10:18
SVPWCS   C       4425   9-03-95  10:18
SVPWIDL  JCL      738  10-03-95  13:44
SVPWSS   C       5660   9-03-95  10:18
USERSVC  ASM   218366   3-03-95   9:52
USERSVC  JCL     2098  10-03-95  13:44
USVCNUM  ASM      656   3-03-95   9:52
        63 file(s)     734140 bytes used

Total files listed:
        68 file(s)     741441 bytes used
                       692736 bytes free
```

A.2 MVS Data Sets

The MVS data sets created during the installation are as follows, where “hlq” is the installation chosen high level qualifier:

Member	**DSN=hlq.DCERACF.ACF**
PTKT	The interface control for passticket
PTST	The interface control for ptst
SVPW	The interface control for SAVEPW client and SAVEPWS server

Member	**DSN=hlq.DCERACF.ASM**
DCERACF	C interface routine used to call the user SVC
DCEKEY	utility to create a new “Secret System Encryption Key”
ICHRCX02	RACF RACHECK post-processing exit
PTKTGEN	C interface routine to RACF Passticket generator
STRMACS	structured assembler macros used in the above
STRREF	structured assembler macro reference manual
USERSVC	a user SVC that implements functions requiring APF authorization by RACF

Member	DSN=hlq.DCERACF.ASM
USVCNUM	specification of the user SVC number chosen by the installation

Member	DSN=hlq.DCERACF.C
DCERACFU	Utility to set and list fields in RACF profiles (User and DCE cross reference)
GETPAC	A subroutine called by the User SVC to get the DCE client's UUID and convert it to a character string to be used to find the associated RACF userid
PTKTCS	Client stub for PTKT server
PTKTLIB	Function library for PTKT clients
PTKTS	Passticket Server
PTKTSS	Server stub for PTKT server
PTSTC	Client for test and demonstration of RACF passtickets
PTSTCS	Client stub for PTST server
PTSTS	Server for test and demonstration of RACF passtickets
PTSTSS	Server stub for PTST server
SAVEPW	Utility to allow an enduser to set and change his or her DCE password (encrypted) stored in his or her RACF user profile This DCE client program can be ported to other platforms.
SAVEPWS	DCE application server (MVS only) for the SAVEPW client to implement the DCE password changes described above
SVPWCS	Client stub for the SVPW interface.
SVPWSS	Server stub for the SVPW interface.

Member	DSN=hlq.DCERACF.H
DCERACF	defines the prototypes of the DCERACF routine, error handling routines, and a function that will enable a program to logon to DCE using the DCE principal name and the encrypted DCE password in the current user's RACF profile.
PTKT	The header file produced by the IDL compiler for the PTKT interface.
PTKTLIB	The header file for PTKT clients
PTST	The header file produced by the IDL compiler for the PTST interface
SVPW	The header file produced by the IDL compiler for the SVPW interface

Member	DSN=hlq.DCERACF.IDL
PTKT	Used by the IDL compiler to produce the PTKT interface used by the passticket client and the PTKTS server programs.
PTST	Used by the IDL compiler to produce the PTST interface used by the PTSTC client and the PTSTS server programs.
SVPW	Used by the IDL compiler to produce the SVPW interface used by the SAVEPW client and the SAVEPWS server programs.

Member	DSN=hlq.DCERACF.JCL
DCECC	Tailored JCL procedure for DCE to assemble

Member	DSN=hlq.DCERACF.JCL
DCEKEY	Assemble and link DCEKEY program
DCEKEYR	Execute the DCEKEY program.
DCELK	Tailored JCL procedure for DCE to link
DCERACF	Assemble and link DCERACF subroutine.
DCERACFU	Compile and link DCERACFU utility program.
GETPAC	Compile and link GETPAC subroutine
ICHRCX02	Assemble and link the RACHECK post-processing exit
ICHRFR01	Assemble and link the RACROUTE table.
ICHRRCDE	Assemble and link the RACF user class table.
PTKTGEN	Assemble PTKTGEN subroutine.
PTKTIDL	Execute the IDL compiler for the PTKT interface. (only included for information)
PTKTLIB	Compile PTKTLIB functions.
PTKTS	Compile and link PTKTS server
PTKTSRUN	Run the PTKTS server
PTSTIDL	Execute the IDL compiler for the PTST interface. (only included for information)
PTSTC	Compile and link PTSTC client
PTSTCRUN	Run the PTSTC client
PTSTS	Compile and link PTSTS server
PTSTSRUN	Run the PTSTS server
SAVEPW	Compile and link the SAVEPW client program.
SAVEPWS	Compile and link the SAVEPWS server program.
SAVEPWSR	Execute the SAVEPWS server program.
SVPWIDL	Execute the IDL compiler for the SVPW interface. (only included for information)
USERSVC	Assemble and link the user SVC.

Member	DSN=hlq.DCERACF.CLIST
DCERACFU	Execute the DCERACFU utility program.
SAVEPW	Execute the SAVEPW client program.

Member	DSN=hlq.DCERACF.OBJ
	Members created during install

Member	DSN= hlq.DCERACF.LOAD
	Members created during install

Appendix B. Source Code

B.1 Installation Utilities

B.1.1 ALLOCATE.CMD

```
/* rexx */
arg session hlq

if length(session) = 0 then do
  say 'You must enter the session to be used'
  say 'Enter session'
  pull session
end

if length(hlq) = 0 then do
  say 'You must enter the high level qualifier to be used'
  say 'Enter hlq'
  pull hlq
end

"send a:\allocate.cli ",
   session":'"hlq".dceracf.allocate.clist' ascii crlf"
```

Figure 9. ALLOCATE.CMD

B.1.2 ALLOCATE.CLIST

```
PROC 3 HLQ VOL UNIT
CONTROL LIST
ALLOC DA('&HLQ..DCERACF.ACF') DIR(20) SPACE(10 10) TRACKS NEW +
  VOL(&VOL) UNIT(&UNIT) RECFM(V B) LRECL(255) BLKSIZE(6240)
ALLOC DA('&HLQ..DCERACF.ASM') DIR(20) SPACE(50 50) TRACKS NEW +
  VOL(&VOL) UNIT(&UNIT) RECFM(F B) LRECL(80) BLKSIZE(6160)
ALLOC DA('&HLQ..DCERACF.C') DIR(20) SPACE(50 50) TRACKS NEW +
  VOL(&VOL) UNIT(&UNIT) RECFM(V B) LRECL(255) BLKSIZE(6240)
ALLOC DA('&HLQ..DCERACF.H') DIR(20) SPACE(10 10) TRACKS NEW +
  VOL(&VOL) UNIT(&UNIT) RECFM(V B) LRECL(255) BLKSIZE(6240)
ALLOC DA('&HLQ..DCERACF.IDL') DIR(20) SPACE(10 10) TRACKS NEW +
  VOL(&VOL) UNIT(&UNIT) RECFM(V B) LRECL(255) BLKSIZE(6240)
ALLOC DA('&HLQ..DCERACF.JCL') DIR(20) SPACE(10 10) TRACKS NEW +
  VOL(&VOL) UNIT(&UNIT) RECFM(F B) LRECL(80) BLKSIZE(6160)
ALLOC DA('&HLQ..DCERACF.CLIST') DIR(20) SPACE(10 10) TRACKS NEW +
  VOL(&VOL) UNIT(&UNIT) RECFM(F B) LRECL(80) BLKSIZE(6160)
ALLOC DA('&HLQ..DCERACF.OBJ') DIR(20) SPACE(10 10) TRACKS NEW +
  VOL(&VOL) UNIT(&UNIT) RECFM(F B) LRECL(80) BLKSIZE(400)
ALLOC DA('&HLQ..DCERACF.LOAD') DIR(20) SPACE(100 100) TRACKS NEW +
  VOL(&VOL) UNIT(&UNIT) RECFM(U)  BLKSIZE(6144)
```

Figure 10. ALLOCATE.CLIST

B.1.3 UPLOAD.CMD

```
/* REXX */

arg session hlq

trace o

/* -------------------- */
/* parse the parameters */
/* -------------------- */

if length(session) = 0 then do
  say 'The host session must be entered'
  say 'Enter session'
  pull session
end

if length(hlq) = 0 then do
  say 'The high level qualify must be entered'
  say 'Enter hlq'
  pull hlq
end

/* -------------- */
/* Load RexxUtils */
/* -------------- */

call RxFuncAdd 'SysLoadFuncs', 'RexxUtil', 'SysLoadFuncs'
call SysLoadFuncs

/* ---------------------------------------- */
/* get list of files in a:\DCERACF directory */
/* ---------------------------------------- */

call SysFileTree 'a:\dceracf\*.*', 'file', 'FO'

do i=1 to file.0

  filename = substr(file.i,lastpos('\',file.i)+1)
  ext = substr(filename,lastpos('.',filename)+1)
  filename = substr(filename,1,lastpos('.',filename)-1)

  /* --------------- */
  /* Upload the file */
  /* --------------- */

  member = filename
  type = ext

  if type = 'CLI' then do
    type = 'CLIST'
  end

  "send a:\DCERACF\"filename"."ext" ",
        session":'"hlq".DCERACF."type"("member")' ASCII CRLF"

end

exit
```

Figure 11. UPLOAD.CMD

B.2 Assembler Routines

The data set hlq.DCERACF.ASM contains the following members:

DCEKEY

DCERACF

ICHRCX02

PTKTGEN

USVCNUM

USERSVC

B.2.1 DCEKEY

```
 TITLE 'DCEKEY'
DCEKEY   START
*--------------------------------------------------------------------*
*
* FUNCTION:
*
*    DCEKEY generates a system secret encryption key that is used
*    to encrypt a user's DCE password. The system secret key is
*    is stored in the RACF database, in a RACF general resource
*    profile in the $DCERACF class. The encrytion key is stored in
*    INSTDATA field of the profile. The naming convention used for
*    the profile/s is:
*
*           <sysid>.CURRENT
*
*         or
*
*           <sysid>.Vnnn
*
*    The utility assumes the following:
*
*    a.  The RACF administrator has created a RACF class of $DCERACF.
*
*    b.  The RACF administrator has created a pool of general
*        resource profiles in that class, conforming to the naming
*        comvention above. The minimum number of profiles is two;
*        one for <sysid>.CURRENT and one for the version number copy
*        of that profile. The version number copy profile is used
*        to save previous keys and enable the decryption of old
*        dce password with re-encryption with the current key when
*        a user next logs into DCE.
*
*    c.  There should be at least one more version number profile
*        than the number of times that this utility has been
*        executed.
*
* ENTRY POINT:
*
*        DCEKEY
*
* ATTRIBUTES:
*
*        REENTRANT
*
* CALLED BY:
*
*        MVS initiator
*
* PARAMETERS:
*
*        none
*
* COPIED MODULES/CONTROL BLOCKS:
*
*        CVT
*        IRRPRXTW
*
* EXIT:
*
*        R15 = 0  -  NORMAL COMPLETION
*
* REGISTER USAGE
*
*        R0   -  WORK
*        R1   -  WORK
```

Figure 12 (Part 1 of 8). Assembler Routine, DCEKEY

```
*        R2   -  WORK
*        R3   -  WORK
*        R4   -  WORK
*        R5   -  WORK
*        R6   -  WORK
*        R7   -  WORK
*        R8   -  WORK
*        R9   -  WORK
*        R10  -  DYNAMIC WORK AREA
*        R11  -  BASE
*        R12  -  RESERVED
*        R13  -  SAVE AREA
*        R14  -  RETURN
*        R15  -  ENTRY POINT
*
* CHANGE ACTIVITY:
*
* Created 1/11/94 Eric Finkelstein
*
*--------------------------------------------------------------------*
 PRINT   GEN
 COPY  STRMACS                          STRUCTURED MACROS
*
*                                 *----------------------------------*
*                                  SAVE REGISTERS
*                                 *----------------------------------*
*
 SAVE (14,12),,DCEKEY.&SYSDATE..&SYSTIME
*
*                                 *----------------------------------*
*                                  ESTABLISH ADDRESSING AND
*                                  CHAIN SAVE AREAS
*                                 *----------------------------------*
*
 USING DCEKEY,11                        ADDRESSING
 USING WORK,10                          ADDRESSING
 LR    11,15                            LOAD BASE
*
*                                 *----------------------------------*
*                                  GET A DYNAMIC WORK AREA
*                                 *----------------------------------*
*
 L     3,=A(WORKE-WORK)                 R3 = L(WORKAREA)
 GETMAIN R,LV=(3)                       GET A WORK AREA
*
*                                 *----------------------------------*
*                                  CLEAR WORK AREA
*                                 *----------------------------------*
*
 LR    4,1                              R4 -> WORK AREA
 L     5,=A(WORKE-WORK)                 R5 =  L(WORK AREA)
 XR    6,6                              R6 =  ZERO
 XR    7,7                              R7 =  ZERO
 MVCL  4,6                              INITITALISE TO NULLS
*
*                                 *----------------------------------*
*                                  CHAIN SAVE AREAS
*                                 *----------------------------------*
*
 ST    1,8(13)                          LSA IN HSA
 ST    13,4(1)                          HSA IN LSA
 LR    13,1                             LSA IN 13
 LR    10,1                             R10 -> WORK
*
*                                 *----------------------------------*
*                                  RETRIEVE CURRENT SECRET KEY
*                                 *----------------------------------*
*
 MVC   ENTITY,=CL256' '                 BLANK ENTITY FIELD
 L     2,16                             R2 -> CVT
```

Figure 12 (Part 2 of 8). Assembler Routine, DCEKEY

```
MVC   ENTITY(8),CVTSNAME-CVTMAP(2)    COPY SYSTEM NAME TO ENTITY
LA    2,ENTITY                        R2 -> ENTITY FIELD
DO    WHILE=(CLI,0(2),NE,C' ')        FIND FIRST BLANK
  LA    2,1(2)                        BUMP R2
ENDDO
MVC   0(8,2),=C'.CURRENT'             MOVE SUFFIX TO ENTITY
MVI   FREEBUF,C'N'                    SET FLAG TO FREE THE BUFFER
MVC   RC,=F'0'                        SET RETURN CODE TO ZERO
MODESET KEY=ZERO,MODE=SUP             SET SUPERVISOR STATE, KEY ZERO
MVC   RACRO1(RACRO1E-RACRO1C),RACRO1C COPY RACROUTE PARAMETERS
RACROUTE REQUEST=EXTRACT,                                          X
              TYPE=EXTRACT,                                        X
              CLASS=CLASS,                                         X
              ENTITY=ENTITY,                                       X
              FIELDS=INSTDATA,                                     X
              SEGMENT=BASE,                                        X
              WORKA=ROUTERWA,                                      X
              RELEASE=1.9.2,                                       X
              MF=(E,RACRO1)
LR    2,1                             SAVE A(RESULT)
ST    2,SAVEBUF                       SAVE A(EXTRACT BUFFER)
IF (LTR,15,15,NZ)                     IF AN ERROR
  ST    15,RC                         SAVE R15
  WTO   'DCEKEY - UNABLE TO EXTRACT ''<SYSID>.CURRENT'' PROFILE',     X
              ROUTCDE=11
  WTO   'DCEKEY - CHECK  1. CLASS $DCERACF IS DEFINED TO RACF   ',    X
              ROUTCDE=11
  WTO   'DCEKEY -        2. CLASS $DCERACF IS DEFINED TO SAF    ',    X
              ROUTCDE=11
  WTO   'DCEKEY -        3. PROFILE HAS BEEN DEFINED TO RACF   ',     X
              ROUTCDE=11
  B     RETURN
ENDIF
MVI   FREEBUF,C'Y'                    SET FLAG TO FREE THE BUFFER
LR    4,2                             R4 -> EXTRACT RESULT
AH    4,EXTWOFF-EXTWKEA(2)            R4 = R4 + OFFSET TO INSTDATA
LH    5,EXTWOFF-EXTWKEA(2)            R5 = L(INSTDATA)
LA    6,RESULT                        R6 -> RESULT
LR    7,5                             R7 = LENGTH TO MOVE
MVCL  6,4                             COPY RESULTS TO WORKAREA
MODESET KEY=NZERO,MODE=PROB           SET PROBLEM STATE, KEY NONZERO
*
*                                  *-----------------------------------*
*                                   IS INSTDATA NOT PRESENT
*                                  *-----------------------------------*
*
 IF    (CLC,KEYLEN,EQ,=F'0')           IF INSTDATA NOT PRESENT
   MVC     KEYVER,=C'000'              INSTVER = 0
 ENDIF
*
*                                  *-----------------------------------*
*                                   INCREMENT VERSION OF KEY
*                                  *-----------------------------------*
*
 PACK  KEYVERP,KEYVER                  PACK THE VERSION NUMBER
 AP    KEYVERP,=PL1'1'                 KEYVERP = KEYVERP + 1
 UNPK  KEYVER,KEYVERP                  UNPACK THE VERSION NUMBER
 OI    KEYVER+2,X'F0'                  REMOVE SIGN
*
*                                  *-----------------------------------*
*                                   GENERATE RANDOM KEY
*                                  *-----------------------------------*
*
 MVC   TIME01(TIME01E-TIME01C),TIME01C  COPY MACRO TO WORK AREA
 TIME  MIC,RANDOM,                                                 X
              ZONE=GMT,                                            X
              DATETYPE=DDMMYYYY,                                   X
              LINKAGE=SYSTEM,                                      X
              MF=(E,TIME01)
 XC    RANDOM(8),RANDOM+8              XOR DATE ON THE TIME
```

Figure 12 (Part 3 of 8). Assembler Routine, DCEKEY

```
 MVI   RANDLEN,X'08'                  SET RANDOM KEY LENGTH
*
*                                    *---------------------------------*
*                                     ENCRYPT THE SECRET KEY INTO INSTDATA
*                                    *---------------------------------*
*
 L     2,16                           R2 -> CVT
 MVI   SYSIDL,X'08'                   SYSIDL = 8
 MVC   SYSIDD(8),CVTSNAME-CVTMAP(2)   SYSID = CVTSNAME
 MODESET KEY=ZERO,MODE=SUP            SET SUPERVISOR STATE, KEY ZERO
 MVC   RACRO2(RACRO2E-RACRO2C),RACRO2C COPY MACRO TO WORK AREA
 RACROUTE REQUEST=EXTRACT,                                            X
               TYPE=ENCRYPT,                                          X
               ENTITY=SYSID,                                          X
               ENCRYPT=(RANDKEY,STDDES),                              X
               WORKA=ROUTERWA,                                        X
               RELEASE=1.9.2,                                         X
               MF=(E,RACRO2)
 IF (LTR,15,15,NZ)                    IF AN ERROR
   ST    15,RC                        SAVE R15
   WTO   'DCEKEY - FAILED TO ENCRYPT SECRET KEY'                      X
               ROUTCDE=11
   B     RETURN
 ENDIF
 MODESET KEY=NZERO,MODE=PROB          SET PROBLEM STATE, KEY NONZERO
*
*                                    *---------------------------------*
*                                     CONVERT TO HEX CHARACTERS
*                                    *---------------------------------*
*
 XR    1,1                            R1 = 0
 LA    2,RANDOM                       R2 -> ENCRYPTED TIME/DATE
 LA    3,KEYKEY                       R3 -> KEYKEY (OUTPUT)
 DO    WHILE=(C,1,NE,=F'4')           DO 4 TIMES
   XR    4,4                          CLEAR R4
   IC    4,0(2)                       R4 = WHOLE BYTE
   SRDL  4,4                          R4 = HI 4 BITS, LOW INTO R5
   IC    4,HEX(4)                     R4 = HEX HI 4 BITS
   STC   4,0(3)                       STORE OUTPUT
   XR    4,4                          CLEAR R4
   SLDL  4,4                          R4 = LOW 4 BITS FROM R5
   IC    4,HEX(4)                     R4 = HEX LOW 4 BITS
   STC   4,1(3)                       STORE OUTPUT
   LA    1,1(1)                       BUMP R1
   LA    2,1(2)                       BUMP R2
   LA    3,2(3)                       BUMP R3
 ENDDO
*
*                                    *---------------------------------*
*                                     COPY TO Vnnn PROFILE
*                                    *---------------------------------*
*
 MVC   ENTITY,=CL256' '               BLANK ENTITY FIELD
 L     2,16                           R2 -> CVT
 MVC   ENTITY(8),CVTSNAME-CVTMAP(2)   COPY SYSTEM NAME TO ENTITY
 LA    2,ENTITY                       R2 -> ENTITY FIELD
 DO    WHILE=(CLI,0(2),NE,C' ')       FIND FIRST BLANK
   LA    2,1(2)                       BUMP R2
 ENDDO
 MVC   0(2,2),=C'.V'                  MOVE SUFFIX TO ENTITY
 LA    2,2(2)                         R2 -> VERSION NO. PORTION
 MVC   0(3,2),KEYVER                  COPY NNN
 MVC   SEGDATA(4),=F'11'              SET THE LENGTH
 MODESET KEY=ZERO,MODE=SUP            SET SUPERVISOR STATE, KEY ZERO
 MVC   RACRO4(RACRO4E-RACRO4C),RACRO4C COPY MACRO TO WORK AREA
 RACROUTE REQUEST=EXTRACT,                                            X
               TYPE=REPLACE,                                          X
               ENTITY=ENTITY,                                         X
               CLASS=CLASS,                                           X
               FIELDS=INSTDATA,                                       X
```

Figure 12 (Part 4 of 8). Assembler Routine, DCEKEY

```
              SEGDATA=SEGDATA,                                        X
              SEGMENT=BASE,                                           X
              WORKA=ROUTERWA,                                         X
              RELEASE=1.9.2,                                          X
              MF=(E,RACRO4)
 IF (LTR,15,15,NZ)                    IF AN ERROR
   ST    15,RC                         SAVE R15
   WTO   'DCEKEY - FAILED TO UPDATE ''<SYSID>.VNNN'' PROFILE',         X
              ROUTCDE=11
   WTO   'DCEKEY - CHECK  1. CLASS $DCERACF IS DEFINED TO RACF   ',    X
              ROUTCDE=11
   WTO   'DCEKEY -        2. CLASS $DCERACF IS DEFINED TO SAF    ',    X
              ROUTCDE=11
   WTO   'DCEKEY -        3. PROFILE HAS BEEN DEFINED TO RACF   ',     X
              ROUTCDE=11
   B     RETURN
 ENDIF
 MODESET KEY=NZERO,MODE=PROB          SET PROBLEM STATE, KEY NONZERO
*
*                                *----------------------------------*
*                                 UPDATE <SYSID>.CURRENT PROFILE
*                                *----------------------------------*
*
 MVC   ENTITY,=CL256' '               BLANK ENTITY FIELD
 L     2,16                           R2 -> CVT
 MVC   ENTITY(8),CVTSNAME-CVTMAP(2)   COPY SYSTEM NAME TO ENTITY
 LA    2,ENTITY                       R2 -> ENTITY FIELD
 DO    WHILE=(CLI,0(2),NE,C' ')       FIND FIRST BLANK
   LA    2,1(2)                       BUMP R2
 ENDDO
 MVC   0(8,2),=C'.CURRENT'            MOVE SUFFIX TO ENTITY
 MVC   SEGDATA(4),=F'11'              SET THE LENGTH
 MODESET KEY=ZERO,MODE=SUP            SET SUPERVISOR STATE, KEY ZERO
 MVC   RACRO3(RACRO3E-RACRO3C),RACRO3C COPY MACRO TO WORK AREA
 RACROUTE REQUEST=EXTRACT,                                            X
              TYPE=REPLACE,                                           X
              ENTITY=ENTITY,                                          X
              CLASS=CLASS,                                            X
              FIELDS=INSTDATA,                                        X
              SEGDATA=SEGDATA,                                        X
              SEGMENT=BASE,                                           X
              WORKA=ROUTERWA,                                         X
              RELEASE=1.9.2,                                          X
              MF=(E,RACRO3)
 IF (LTR,15,15,NZ)                    IF AN ERROR
   ST    15,RC                         SAVE R15
   WTO   'DCEKEY - FAILED TO UPDATE ''<SYSID>.CURRENT'' PROFILE',      X
              ROUTCDE=11
   WTO   'DCEKEY - CHECK  1. CLASS $DCERACF IS DEFINED TO RACF   ',    X
              ROUTCDE=11
   WTO   'DCEKEY -        2. CLASS $DCERACF IS DEFINED TO SAF    ',    X
              ROUTCDE=11
   WTO   'DCEKEY -        3. PROFILE HAS BEEN DEFINED TO RACF   ',     X
              ROUTCDE=11
   B     RETURN
 ENDIF
 MODESET KEY=NZERO,MODE=PROB          SET PROBLEM STATE, KEY NONZERO
*
*-------------------------------------------------------------------*
*                        RETURN
*-------------------------------------------------------------------*
RETURN EQU *
*
*                                *----------------------------------*
*                                 FREE THE EXTRACT BUFFER
*                                *----------------------------------*
*
 IF    (CLI,FREEBUF,EQ,C'Y')          DO WE HAVE A BUFFER
   MODESET KEY=ZERO,MODE=SUP          SET SUPERVISOR STATE, KEY ZERO
   L   2,SAVEBUF                      R2 -> EXTRACT BUFFER
```

Figure 12 (Part 5 of 8). Assembler Routine, DCEKEY

```
   XR  3,3                           ZERO OUT R3
   XR  4,4                           ZERO OUT R4
   USING EXTWKEA,2                   BASE THE AREA ON R2
   ICM 3,1,EXTWSP                    MOVE THE AREA SUBPOOL INTO R3
   ICM 4,7,EXTWLN                    MOVE THE AREA LENGTH INTO R4
   DROP 2                            DROP BASING ON REGISTER 2
   FREEMAIN R,LV=(4),A=(2),SP=(3)    FREE STORAGE BEFORE PROCESSING
   MODESET KEY=NZERO,MODE=PROB       SET PROBLEM STATE, KEY NONZERO
 ENDIF
 IF   (CLC,RC,EQ,=F'0')
   WTO 'DCEKEY - NEW SYSTEM SECRET KEY CREATED SUCCESSFULLY',          x
               ROUTCDE=11
 ENDIF
 L     8,RC                        R8 = RC
 L     13,4(13)                    R13 -> HSA
 L     3,=A(WORKE-WORK)            R3 = L(WORKAREA)
 FREEMAIN R,LV=(3),A=(10)          FREE WORK AREA
 LR    15,8                        R15 = RC
 RETURN (14,12),RC=(15)            RETURN
*
*-------------------------------------------------------------------*
 TITLE 'CONSTANTS'
*-------------------------------------------------------------------*
*
CONSTANT DS    OD
HEX      DC    C'0123456789ABCDEF'
INSTDATA DC    A(1)                NUMBER OF FIELDS
         DC    CL8'INSTDATA'       INSTDATA FIELD
BASE     DC    CL8'BASE'           SEGMENT BASE
CLASS    DC    CL8'$DCERACF'       DCEKEY CLASS
TIME01C  TIME  LINKAGE=SYSTEM,MF=L
TIME01E  EQU   *
RACR01C  RACROUTE REQUEST=EXTRACT,                                  X
               TYPE=EXTRACT,                                        X    X
               CLASS=*-*,                                           X
               ENTITY=*-*,                                          X
               FIELDS=*-*,                                          X
               SEGMENT=*-*,                                         X
               WORKA=*-*,                                           X
               RELEASE=1.9.2,                                       X
               MF=L
RACR01E  EQU   *
RACR02C  RACROUTE REQUEST=EXTRACT,                                  X
               TYPE=ENCRYPT,                                        X    X
               ENCRYPT=(*-*,STDDES),                                X
               ENTITY=*-*,                                          X
               WORKA=*-*,                                           X
               RELEASE=1.9.2,                                       X
               MF=L
RACR02E  EQU   *
RACR03C  RACROUTE REQUEST=EXTRACT,                                  X
               TYPE=REPLACE,                                        X    X
               ENTITY=*-*,                                          X
               CLASS=*-*,                                           X
               FIELDS=*-*,                                          X
               SEGDATA=*-*,                                         X
               SEGMENT=*-*,                                         X
               WORKA=*-*,                                           X
               RELEASE=1.9.2,                                       X
               MF=L
RACR03E  EQU   *
RACR04C  RACROUTE REQUEST=EXTRACT,                                  X
               TYPE=REPLACE,                                        X    X
               ENTITY=*-*,                                          X
               CLASS=*-*,                                           X
               FIELDS=*-*,                                          X
               SEGDATA=*-*,                                         X
               SEGMENT=*-*,                                         X
               WORKA=*-*,                                           X
               RELEASE=1.9.2,                                       X
```

Figure 12 (Part 6 of 8). Assembler Routine, DCEKEY

```
              MF=L
RACRO4E  EQU   *
*
*--------------------------------------------------------------------*
 TITLE 'LITERALS'
*--------------------------------------------------------------------*
*
 LTORG
*
*--------------------------------------------------------------------*
 TITLE 'DYNAMIC WORK AREA'
*--------------------------------------------------------------------*
*
WORK     DSECT
LSA      DS    18F                   LSA
FREEBUF  DS    CL1                   FREE EXTRACT BUFFER FLAG
SAVEBUF  DS    F                     SAVED A(EXTRACT BUFFER)
RC       DS    F                     RETURN CODE
ENTITY   DS    CL256                 ENTITY FOR REQUEST
RESULT   DS    CL256                 INSTDATA
         ORG   RESULT
SEGDATA  DS    OCL256
KEYLEN   DS    A(*-*)                LENGTH OF RESULT
KEYKEY   DS    CL8                   SECRET KEY
KEYVER   DS    CL3                   SECRET KEY VERSION
         ORG
KEYVERP  DS    PL2
TIMEO1   TIME  LINKAGE=SYSTEM,MF=L
*
RANDKEY  DS    OCL9
RANDLEN  DS    AL1
RANDOM   DS    CL16
*
SYSID    DS    OCL9
SYSIDL   DS    AL1
SYSIDD   DS    CL8
*
RACRO1   RACROUTE REQUEST=EXTRACT,                                  X
               TYPE=EXTRACT,                                        X    X
               CLASS=*-*,                                           X
               ENTITY=*-*,                                          X
               FIELDS=*-*,                                          X
               SEGMENT=*-*,                                         X
               WORKA=*-*,                                           X
               RELEASE=1.9.2,                                       X
               MF=L
RACRO2   RACROUTE REQUEST=EXTRACT,                                  X
               TYPE=ENCRYPT,                                        X    X
               ENCRYPT=(*-*,STDDES),                                X
               ENTITY=*-*,                                          X
               WORKA=*-*,                                           X
               RELEASE=1.9.2,                                       X
               MF=L
RACRO3   RACROUTE REQUEST=EXTRACT,                                  X
               TYPE=REPLACE,                                        X    X
               ENTITY=*-*,                                          X
               CLASS=*-*,                                           X
               FIELDS=*-*,                                          X
               SEGDATA=*-*,                                         X
               SEGMENT=*-*,                                         X
               WORKA=*-*,                                           X
               RELEASE=1.9.2,                                       X
               MF=L
RACRO4   RACROUTE REQUEST=EXTRACT,                                  X
               TYPE=REPLACE,                                        X    X
               ENTITY=*-*,                                          X
               CLASS=*-*,                                           X
               FIELDS=*-*,                                          X
               SEGDATA=*-*,                                         X
               SEGMENT=*-*,                                         X
```

Figure 12 (Part 7 of 8). Assembler Routine, DCEKEY

```
              WORKA=*-*,                                              X
              RELEASE=1.9.2,                                          X
              MF=L
ROUTERWA DS   OD,512X
WORKE    DS   OD                    END OF AREA
*
*-------------------------------------------------------------------*
 TITLE 'DSECTS'
*-------------------------------------------------------------------*
*
 CVT DSECT=YES
 IRRPRXTW
 END
```

Figure 12 (Part 8 of 8). Assembler Routine, DCEKEY

B.2.2 DCERACF

```
 TITLE 'DCERACF'
DCERACF  START
*--------------------------------------------------------------------*
*
* FUNCTION:
*
*        This is an interface routine between a 'C' caller and the
*        DCE/RACF user SVC. It is 'C' conformant and passes the
*        'C' environment registers 12 and 13 through to the user
*        SVC as well as the caller's parameters.
*
* ENTRY POINT:
*
*        DCERACF
*
* ATTRIBUTES:
*
*        REENTRANT
*
* CALLED BY:
*
*        'C' DCE SERVERS
*
* PARAMETERS:
*
*        none
*
* COPIED MODULES/CONTROL BLOCKS:
*
*        CVT
*        IRRPRXTW
*
* EXIT:
*
*        R15 = 0  -  NORMAL COMPLETION
*
* REGISTER USAGE
*
*        R0   -  WORK
*        R1   -  WORK
*        R2   -  WORK
*        R3   -  WORK
*        R4   -  WORK
*        R5   -  WORK
*        R6   -  WORK
*        R7   -  WORK
*        R8   -  WORK
*        R9   -  WORK
*        R10  -  DYNAMIC STORAGE AREA
*        R11  -  BASE
*        R12  -  RESERVED FOR 'C' ENVIRONMENT
*        R13  -  SAVE AREA/'C' DSA
*        R14  -  RETURN
*        R15  -  ENTRY POINT
*
* CHANGE ACTIVITY:
*
* Created 1/11/94 Eric Finkelstein
*
*--------------------------------------------------------------------*
 PRINT   GEN
 COPY  STRMACS                        STRUCTURED MACROS
 COPY  USVCNUM                        SVC DETAILS
*
*                                *----------------------------------*
```

Figure 13 (Part 1 of 2). Assembler Routine, DCERACF

```
*                                C PROLOG
*                               *---------------------------------*
*
 EDCPRLG BASEREG=11,USRDSAL=USRSIZE
 USING DSA,13                        ESTABLISH ADDRESSABILITY
 LR 8,1                              SAVE R1
*
*                               *---------------------------------*
*                                CALL THE USER SVC
*                               *---------------------------------*
*
 ST    12,CENVR12                    SAVE R12 TO PASS TO THE SVC
 ST    13,CENVR13                    SAVE R13 TO PASS TO THE SVC
 LA    15,CENV                       R15 -> CENV
 LR    0,15                          RO -> CENV
 LR    1,8                           R1 -> CALLER'S PARAMETER LIST
 SVC   &USERSVC                      CALL THE USER SVC
 ST    15,RC                         SAVE THE RETURN CODE
*
*                               *---------------------------------*
*                                RETURN
*                               *---------------------------------*
*
 L     15,RC                         R15 = RC
 EDCEPIL                             RETURN
*
*--------------------------------------------------------------------*
 TITLE 'CONSTANTS'
*--------------------------------------------------------------------*
*
CONSTANT DS    OD
USRSIZE   EQU   USRDSAE-USRDSA
*
*--------------------------------------------------------------------*
 TITLE 'LITERALS'
*--------------------------------------------------------------------*
*
 LTORG
*
*--------------------------------------------------------------------*
 TITLE 'DYNAMIC STORAGE AREA'
*--------------------------------------------------------------------*
*
DSA      EDCDSAD
USRDSA   DS    OF
RC       DS    F                     RETURN CODE
*
CENV     DS    OF                    USER SVC PARAMETER LIST
CENVR12  DS    F                     C ENVIRONMENT R12
CENVR13  DS    F                     C ENVIRONMENT R13
*
*                               *---------------------------------*
USRDSAE  DS    OD                     END OF DSA
*                               *---------------------------------*
*
*--------------------------------------------------------------------*
 TITLE 'DSECTS'
*--------------------------------------------------------------------*
*
 END
```

Figure 13 (Part 2 of 2). Assembler Routine, DCERACF

B.2.3 ICHRCX02

```
 TITLE 'ICHRCX02'
ICHRCX02 START
*--------------------------------------------------------------------*
*
* FUNCTION:
*
*    ICHRCX02 is the RACHECK post-processing exit. It is used here
*    to close a security hole introduced by the RACF and DCE Security
*    inter-operation code.
*
*    The user SVC provided, assumed that the binding handle for
*    the client session was opaque, meaning that the contents were
*    secured (that is encrypted)  by DCE runtime. This meaning was
*    incorrect.
*
*    Opaque only means that there is no mapping provided for the
*    structure.  The contents have already been decrypted by the
*    DCE runtime and it could be modified by a rogue non-APF
*    authorized server.
*
*    For example, it could be searched for the client's UUID and
*    replaced with another UUID. This UUID is used by the user SVC
*    to logon the client to RACF without a password because the
*    user has already been authenticated by DCE security. That way the
*    rogue server could get access to resources to which the
*    client had access without have access itself.
*
*    To close the integrity hole for a non-APF authorized DCE
*    server, this exit forces any RACHECK for an ACEE pointed to
*    by TCBSENV, which has the ACEEAPLN field in the ACEE
*    containing 'DCERACF',  to be re-driven to also check against
*    the address space ASXBSENV ACEE, if the TCBSENV ACEE was
*    successful.
*
*    This means that the client can only have access to resources
*    to which the DCE application server also has access.
*
*    While this is a restriction, it still allows the DCE server
*    on MVS to discriminate between clients for the resources to
*    which does server does have access and have the RACF logging
*    and auditing controls.
*
*    NOTE: If the installation already uses the ICHRCX02 exit,
*    this code will have to be merged with it, to ensure the
*    integrity exposure is closed. This code will only process the
*    request if the field ACEEAPLN (the application name) in the
*    ACEE is DCERACF and the job step is non-APF authorised.
*    Therefore, only ACEEs created by the user SVC will be
*    effected by this code.
*
* ENTRY POINT:
*
*        ICHRCX02
*
* ATTRIBUTES:
*
*        REENTRANT, AMODE 31, RMODE ANY
*
* CALLED BY:
*
*        RACF
*
* PARAMETERS:
*
*        ICHRCXP
```

Figure 14 (Part 1 of 4). Assembler Routine, ICHRCX02

```
*
* COPIED MODULES/CONTROL BLOCKS:
*
*        CVT
*        IRRPRXTW
*
* EXIT:
*
*        R15 = 0  -  NORMAL COMPLETION
*
* REGISTER USAGE
*
*        R0   -  WORK
*        R1   -> parameter list
*        R2   -> ICHRCXP (SAVED)
*        R3   -  WORK
*        R4   -  WORK
*        R5   -  WORK
*        R6   -  WORK
*        R7   -  WORK
*        R8   -  WORK
*        R9   -  WORK
*        R10  -  DYNAMIC WORK AREA
*        R11  -  BASE
*        R12  -  RESERVED
*        R13  -  SAVE AREA
*        R14  -  RETURN
*        R15  -  ENTRY POINT
*
* CHANGE ACTIVITY:
*
* Created 11/2/95 Eric Finkelstein
*
*--------------------------------------------------------------------*
 PRINT   GEN
ICHRCX02 AMODE 31
ICHRCX02 RMODE ANY
 COPY  STRMACS                          STRUCTURED MACROS
*
*                                  *-----------------------------------*
*                                   SAVE REGISTERS
*                                  *-----------------------------------*
*
 SAVE (14,12),,ICHRCX02.&SYSDATE..&SYSTIME
*
*                                  *-----------------------------------*
*                                   ESTABLISH ADDRESSING AND
*                                   CHAIN SAVE AREAS
*                                  *-----------------------------------*
*
 USING ICHRCX02,11                      ADDRESSING
 USING WORK,10                          ADDRESSING
 LR    11,15                            LOAD BASE
 LR    2,1                              SAVE A(PARAMETER LIST) IN R2
*
*                                  *-----------------------------------*
*                                   GET A DYNAMIC WORK AREA
*                                  *-----------------------------------*
*
 L     3,=A(WORKE-WORK)                 R3 = L(WORKAREA)
 GETMAIN R,LV=(3)                       GET A WORK AREA
*
*                                  *-----------------------------------*
*                                   CLEAR WORK AREA
*                                  *-----------------------------------*
*
 LR    4,1                              R4 -> WORK AREA
 L     5,=A(WORKE-WORK)                 R5 =  L(WORK AREA)
 XR    6,6                              R6 =  ZERO
 XR    7,7                              R7 =  ZERO
```

Figure 14 (Part 2 of 4). Assembler Routine, ICHRCX02

```
 MVCL  4,6                          INITITALISE TO NULLS
*
*                                 *---------------------------------*
*                                  CHAIN SAVE AREAS
*                                 *---------------------------------*
*
 ST    1,8(13)                     LSA IN HSA
 ST    13,4(1)                     HSA IN LSA
 LR    13,1                        LSA IN 13
 LR    10,1                        R10 -> WORK
*
*                                 *---------------------------------*
*                                  IS THE CALLER NON-APF AUTHORIZED
*                                 *---------------------------------*
*
 TESTAUTH FCTN=1                   CHECK APF AUTHORIZATION
 IF (LTR,15,15,NZ)                 IF R15 > 0 THEN NON-APF AUTH
*
*                                 *---------------------------------*
*                                  FIND CURRENT TCB
*                                 *---------------------------------*
*
   L 3,16                          R3 -> CVT
   L 3,CVTTCBP-CVTMAP(3)           R3 -> A(NEXT TCB, CURRENT TCB)
   L 3,4(3)                        R3 -> CURRENT TCB
*
*                                 *---------------------------------*
*                                  FIND ACEE (TCB OR ASXB)
*                                 *---------------------------------*
*
   L 3,TCBSENV-TCB(3)              R3 -> ACEE
*
*                                 *---------------------------------*
*                                  IF THERE IS AN ACEE FOR THE TCB
*                                 *---------------------------------*
*
   IF (LTR,3,3,NZ)                 IF R3 NON-ZERO
*
*                                 *---------------------------------*
*                                  IF ACEE CREATED BY THE DCERACF SVC
*                                 *---------------------------------*
*
     IF (CLC,ACEEAPLN-ACEE(8,3),EQ,=CL8'DCERACF')
*
*                                 *---------------------------------*
*                                  IF RACHECK RETURN CODE WILL BE ZERO
*                                 *---------------------------------*
*
       L 6,RCXRCODE-RCXPL(2)       R3 -> RACHECK RETURN CODE
       IF (CLC,0(4,6),EQ,=F'0')    IF RACHECK RETURN CODE ZERO
*
*                                 *---------------------------------*
*                                  IF NOT ALREADY RE-DRIVEN
*                                 *---------------------------------*
*
         IF (CLC,RCXWA-RCXPL(4,2),NE,=CL4'DCER')
           L 4,16                  R4 -> CVT
           L 4,CVTTCBP-CVTMAP(4)   R4 -> A(NEXT TCB, CURRENT TCB)
           L 4,12(4)               R4 -> CURRENT ASCB
           L 4,ASCBASXB-ASCB(4)    R4 -> ASXB
           L 4,ASXBSENV-ASXB(4)    R4 -> ASXB ACEE
           L 5,RCXACEE-RCXPL(2)    R5 -> ACEE POINTER
           ST 4,0(5)               FORCE USE OF ASXB ACEE
           MVC RCXWA-RCXPL(4,2),=CL4'DCER' INDICATE A RE-DRIVE
           MVC RC,=F'4'            FORCE RE-DRIVE
         ENDIF
       ENDIF
     ENDIF
   ENDIF
 ENDIF
```

Figure 14 (Part 3 of 4). Assembler Routine, ICHRCX02

```
*
*-------------------------------------------------------------------*
*                        RETURN
*-------------------------------------------------------------------*
 L     8,RC                           R8 = RC
 L     13,4(13)                       R13 -> HSA
 L     3,=A(WORKE-WORK)               R3 = L(WORKAREA)
 FREEMAIN R,LV=(3),A=(10)             FREE WORK AREA
 LR    15,8                           R15 = RC
 RETURN (14,12),RC=(15)               RETURN
*
*
*-------------------------------------------------------------------*
 TITLE 'CONSTANTS'
*-------------------------------------------------------------------*
*
CONSTANT DS    0D
*
*-------------------------------------------------------------------*
 TITLE 'LITERALS'
*-------------------------------------------------------------------*
*
 LTORG
*
*-------------------------------------------------------------------*
 TITLE 'DYNAMIC WORK AREA'
*-------------------------------------------------------------------*
*
WORK     DSECT
LSA      DS    18F                    LSA
RC       DS    F                      RETURN CODE
WORKE    DS    0D
*
*-------------------------------------------------------------------*
 TITLE 'DSECTS'
*-------------------------------------------------------------------*
*
 CVT DSECT=YES
 IHAACEE
 ICHRCXP
 IKJTCB
 IHAASCB
 IHAASXB
 END
```

Figure 14 (Part 4 of 4). Assembler Routine, ICHRCX02

B.2.4 PTKTGEN

```
 TITLE 'PTKTGEN'
PTKTGEN  START
*-------------------------------------------------------------------*
*
* FUNCTION:
*
*      This is an interface routine between a 'C' caller and the
*      MVS callable service to generate passtickets.
*
*      The APPNAME used for the call is built from the SMF system
*      identification (SID parameter in member IEASMF00 of
*      'SYS1.PARMLIB' with special characters removed with a prefix
*      of 'DCE'.  For example, SID=S*Y5 would give a APPNAME of
*      'DCESY5'.
*
* ENTRY POINT:
*
*        PTKTGEN
*
* ATTRIBUTES:
*
*        REENTRANT
*
* CALLED BY:
*
*        'C' DCE SERVERS
*
* PARAMETERS:
*
*        USERID - RACF userid (null terminated string) (input)
*
*        TICKET  - generated ticket (null terminated string) (output)
*
* COPIED MODULES/CONTROL BLOCKS:
*
*        CVT
*        IEESMCA
*
* EXIT:
*
*        R15 = 0  -  NORMAL COMPLETION
*
* REGISTER USAGE
*
*        R0   -  WORK
*        R1   -  WORK
*        R2   -  WORK
*        R3   -  WORK
*        R4   -  WORK
*        R5   -  WORK
*        R6   -  WORK
*        R7   -  WORK
*        R8   -  WORK
*        R9   -  WORK
*        R10  -  DYNAMIC STORAGE AREA
*        R11  -  BASE
*        R12  -  RESERVED FOR 'C' ENVIRONMENT
*        R13  -  SAVE AREA/'C' DSA
*        R14  -  RETURN
*        R15  -  ENTRY POINT
*
* CHANGE ACTIVITY:
*
* Created 18/2/95 Eric Finkelstein
*
```

Figure 15 (Part 1 of 4). Assembler Routine, PTKTGEN

```
*--------------------------------------------------------------------*
 PRINT   GEN
 COPY  STRMACS                          STRUCTURED MACROS
*
*                                *----------------------------------*
*                                 C PROLOG
*                                *----------------------------------*
*
 EDCPRLG BASEREG=11,USRDSAL=USRSIZE
 USING DSA,13                           ESTABLISH ADDRESSABILITY
 LR 8,1                                 SAVE R1
 MVI DBG,C'N'                           TURN DEBUG MESSAGES ON/OFF
*
*                                *----------------------------------*
*                                 COPY USERID
*                                *----------------------------------*
*
 XR 1,1                                 R1 = 0 (COUNT)
 L 2,0(8)                               R2 -> CALLER'S USERID PARAMETER
 LA 3,USERID+1                          R3 -> USERID DATA
 MVC USERID,=CL9' '                     CLEAR USERID
 DO WHILE=(CLI,0(2),NE,X'00')           DO WHILE NOT NULL
   MVC 0(1,3),0(2)                      COPY CHARACTER
   OI  0(3),C' '                        FORCE UPPER CASE
   LA 1,1(1)                            BUMP COUNT
   LA 2,1(2)                            BUMP INPUT
   LA 3,1(3)                            BUMP OUTPUT
 ENDDO
 STC   1,USERID                         STORE LENGTH OF USERID
*
*                                *----------------------------------*
*                                 BUILD APPNAME
*                                *----------------------------------*
*
 L 2,16                                 R2 -> CVT
 L 2,CVTSMCA-CVT(2)                     R2 -> SMCA
 LA 2,SMCASID-SMCABASE(2)               R2 -> SMF SYSTEM ID
 MVC APPNAME+1(8),=CL8'DCE'             INITILISE APPNAME
 LA 3,APPNAME+4                         R3 -> SID IN APPNAME
 L 1,=F'4'                              R1 = LOOP COUNT
 DO WHILE=(LTR,1,1,NZ)
   IF (CLI,0(2),GE,C'A')                IF  NOT A SPECIAL CHARACTER
     MVC 0(1,3),0(2)                    COPY CHARACTER
     LA 3,1(3)                          BUMP OUTPUT
   ENDIF
   S 1,=F'1'                            DECREMENT COUNT
   LA 2,1(2)                            BUMP INPUT
 ENDDO
 XR 1,1                                 R1 = 0 (COUNT)
 LA 2,APPNAME+1                         R2 -> APPNAME DATA
 DO WHILE=(CLI,0(2),NE,C' ')            DO WHILE NOT BLANK
   LA 1,1(1)                            BUMP COUNT
   LA 2,1(2)                            BUMP INPUT
 ENDDO
 STC 1,APPNAME                          STORE COUNT
*
*                                *----------------------------------*
*                                 CALL PASSTICKET GENERATOR
*                                *----------------------------------*
*
 MODESET KEY=ZERO,MODE=SUP              SET SUPERVISOR STATE, KEY ZERO
 LA 2,USERID                            R2 -> USERID
 LA 3,APPNAME                           R3 -> APPNAME
 STM 2,3,PLIST                          SAVE IN PLIST
 OI PLIST+4,X'80'                       SET END OF LIST
*
 L 2,16                                 R2 -> CVT
 L 2,CVTRAC-CVT(2)                      R2 -> RCVT
 L 15,RCVTPTGN-RCVT(2)                  R15 -> PASSTICKET GENERATOR
*
```

Figure 15 (Part 2 of 4). Assembler Routine, PTKTGEN

```
 LA 1,PLIST                          R1 -> PLIST
 CALL (15)                           CALL PASSTICKET GENERATOR
*
 ST 15,RC                            SAVE RETURN CODE
 STM 0,1,TICKET                      SAVE GENERATED TICKET
 LM 0,1,=D'0'                        DON'T LEAVE TICKET AROUND
 MODESET KEY=NZERO,MODE=PROB         SET PROBLEM STATE, KEY NONZERO
*
 IF (CLI,DBG,EQ,C'Y')
   MVC WTO(WTOE-WTOC),WTOC           COPY WTO LIST TO DSA
   MVC WTO+4(7),=C'USERID='
*
   LA 3,WTO+11
   LA 2,USERID+1                     R2 -> USERID
   MVC 0(8,3),0(2)                   COPY CHARACTER
*
   LA 3,WTO+20
   MVC 0(8,3),=C'USERLEN='
*
   LA 3,WTO+28
   XR 1,1
   IC 1,USERID
   IC 1,HEX(1)
   STC 1,0(3)
*
   LA 3,WTO+30
   MVC 0(8,3),=C'APPNAME='
*
   LA 3,WTO+38
   LA 2,APPNAME+1                    R2 -> APPLNAME
   MVC 0(8,3),0(2)                   COPY CHARACTER
*
   LA 3,WTO+47
   MVC 0(7,3),=C'APPLEN='
*
   LA 3,WTO+54
   XR 1,1
   IC 1,APPNAME
   IC 1,HEX(1)
   STC 1,0(3)
*
   WTO MF=(E,WTO)
 ENDIF
*
*                               *-----------------------------------*
*                                COPY TICKET
*                               *-----------------------------------*
*
 L 2,4(8)                            R2 -> CALLER'S TICKET PARAMETER
 MVC 0(9,2),=XL9'00'                 CLEAR CALLER'S PARAMETER
 MVC 0(8,2),TICKET                   COPY TICKET
 MVC TICKET,=XL9'00'                 DON'T LEAVE TICKET AROUND
*
*                               *-----------------------------------*
*                                RETURN
*                               *-----------------------------------*
*
 L 15,RC                             R15 = RC
 EDCEPIL                             RETURN
*
*-------------------------------------------------------------------*
 TITLE 'CONSTANTS'
*-------------------------------------------------------------------*
*
CONSTANT DS    0D
USRSIZE   EQU   USRDSAE-USRDSA
HEX      DC    C'0123456789ABCDEF'
WTOC     WTO   '                                                    X
                                                                    X
                          ',ROUTCDE=(8,11),DESC=(7),MF=L
```

Figure 15 (Part 3 of 4). Assembler Routine, PTKTGEN

```
WTOE     EQU   *
*
*-------------------------------------------------------------------*
 TITLE 'LITERALS'
*-------------------------------------------------------------------*
*
 LTORG
*
*-------------------------------------------------------------------*
 TITLE 'DYNAMIC STORAGE AREA'
*-------------------------------------------------------------------*
*
DSA      EDCDSAD
USRDSA   DS    0F
RC       DS    F                      RETURN CODE
TICKET   DS    CL9                    TICKET
USERID   DS    CL9                    USERID
APPNAME  DS    CL9                    APPNAME
DBG      DS    CL1
PLIST    DS    2F                     PARAMETER LIST
WTO      WTO   '                                                      X
                                                                      X
                           ',ROUTCDE=(8,11),DESC=(7),MF=L
*
*                                *-----------------------------------*
USRDSAE  DS    0D                  END OF DSA
*                                *-----------------------------------*
*
*-------------------------------------------------------------------*
 TITLE 'DSECTS'
*-------------------------------------------------------------------*
*
 CVT DSECT=YES
 ICHPRCVT
 IEESMCA
 END
```

Figure 15 (Part 4 of 4). Assembler Routine, PTKTGEN

B.2.5 USVCNUM

```
*--------------------------------------------------------------------*
*  DCE/RACF USER SVC DETAILS
*--------------------------------------------------------------------*
          GBLC  &USERSVC
          GBLC  &USERSVCM
&USERSVC  SETC  '255'               USER SVC NUMBER
&USERSVCM SETC  'IGC0025E'          USER SVC MODULE NAME
*
```

Figure 16. Assembler Routine, USVCNUM

B.2.6 USERSVC

```
 TITLE 'DCE/RACF USER SVC'
*-------------------------------------------------------------------*
*
* FUNCTION:
*
*    This user SVC provides access to authorised RACF services as
*    a part of the RACF/DCE interoperation. The functions it provides
*    are described in the parameter section below.
*
* ENTRY POINT:
*
*        Set by the &USERSVCM variable in USVCNUM parameter member
*
* ATTRIBUTES:
*
*        Reentrant, Refreshable, Supervisor state, key zero
*
* CALLED BY:
*
*        DCERACF - assembler interface from DCE 'C' server
*
* PARAMETERS:
*
*    R0 -> 'C' environment (caller's R12, and R13)
*
*    R1 -> standard parameter list dependant on the function
*          requested where the requested function is in the first
*          parameter.
*
*    Note: DCERACF in ERICFIN.DCE.H contains 'C' defines for the
*          REQUEST values as follows:
*
*         #define RACF_logon                  1
*         #define RACF_check_authorisation    2
*         #define RACF_logoff                 3
*         #define RACF_set_DCE_password       4
*         #define RACF_get_DCE_password       5
*         #define RACF_get_DCE_principal      6
*         #define RACF_set_DCE_principal      7
*         #define RACF_get_DCE_usrdata        8
*         #define RACF_get_MVS_sysid          9
*         #define RACF_set_DCE_uuid          10
*         #define RACF_get_DCE_uuid          11
*
*    Logon to RACF
*    -------------
*
*      REQUEST = F'1' - RACF_logon
*      BH      - binding handle (opaque).
*
*                The binding handle passed by the DCE application
*                server enables the GETPAC subroutine to
*                communicate with the DCE Security Server to
*                find the RACF userid associated with this principal.
*
*      PASSTCK - optional passticket to be used to authenticate the
*                logon request (null terminated string)
*
*                If this parameter is present, the PASSTCK is
*                assumed be either the clear password for RACF
*                userid, or a one time use passticket generated by
*                the "PASSTICKET" server. If the passticket or
*                password is accepted by RACF, there are no
*                restrictions to the client's access to RACF
*                resources to which he/she has been authorised.
```

Figure 17 (Part 1 of 39). Assembler Routine, USERSVC

```
*
*               If, however, the parameter is not present, the
*               RACF userid under which the DCE application
*               server is executing MUST also have at least the
*               requested level of access (implicity requested or
*               explicitly requested) as the client. This
*               restriction is placed to close an integrity
*               exposure introduced by this user SVCs use of the
*               unauthenticated binding handle to get the RACF
*               userid.
*
*    Check authorisation to a resource
*    ---------------------------------
*
*      REQUEST = F'2' - RACF_check_authorisation
*      BH      - binding handle (opaque)
*      CLASS   - class of the resource (null terminated string)
*      ENTITY  - resource to be tested (null terminated string)
*      ACCESS  - access requested (null terminated string)
*
*                'READ'
*                'WRITE'
*                'CONTROL'
*                'ALTER'
*
*
*    Logoff RACF
*    -----------
*
*      REQUEST = F'3' - RACF_logoff
*      BH      - binding handle (opaque)
*
*
*    Set encrypted DCE password
*    --------------------------
*
*      REQUEST  = F'4' - RACF_set_DCE_password
*      PASSWD   - clear text to be encrypted and written into
*                 the user's RACF profile and ACEE
*                 (null terminated string) (input)
*
*    Get DCE password
*    ----------------
*
*      REQUEST = F'5' - RACF_get_DCE_password
*      PASSWD  - decrypted DCE password (null terminated string)
*                (output)
*
*    Get DCE principal
*    -----------------
*
*      REQUEST  = F'6' - RACF_get_DCE_principal from current ACEE
*      PRINCIPAL - DCE principal name (null terminated string)
*                 (output)
*
*    Set DCE principal
*    -----------------
*
*      The caller must have ALTER access to 'ADMINISTRATOR' profile
*      in class $DCERACF.
*
*      REQUEST  = F'7' - RACF_set_DCE_principal in profile
*      USERID   - RACF userid (null terminated string) (input)
*      PRINCIPAL - DCE principal name (null terminated string)
*                 (input)
*
*    List USRDATA
*    ------------
*
*      The caller must have ALTER access to 'ADMINISTRATOR' profile
```

Figure 17 (Part 2 of 39). Assembler Routine, USERSVC

```
*      in class $DCERACF.
*
*      REQUEST  = F'8' - RACF_get_DCE_usrdata in profile
*      USERID   - RACF userid (null terminated string) (input)
*      FORMAT   - Format of the output. 'C' for character, anything
*                 else for hex.
*      WORKAREA - contents of USRDATA field of the requested userid
*                 (null terminated string) (output)
*
*    Get RACF sysid
*    --------------
*
*      REQUEST  = F'9' - RACF_get_MVS_sysid
*      SYSID    - System id from MVS CVT
*                 (null terminated string) (output)
*
*    Set DCE uuid profile
*    --------------------
*
*      REQUEST  = F'10' - RACF_set_DCE_uuid
*      UUID     - UUID string (input) (null terminated)
*      USERID   - associated RACF userid (input) (null terminated)
*
*    Get DCE uuid profile
*    --------------------
*
*      REQUEST  = F'11' - RACF_get_DCE_uuid
*      UUID     - UUID string (input) (null terminated)
*      USERID   - associated RACF userid (output) (null terminated)
*
*
* COPIED MODULES/CONTROL BLOCKS:
*
*       CVT
*       IRRPRXTW
*       IKJTCB  - TCB
*       IHAASCB - ASCB
*       IHAASXB - ASXB
*       IHAACEE - ACEE
*
* RACF PROFILE FIELDS USED:
*
*       System secret encryption key profiles.
*       --------------------------------------
*
*          CLASS:     $DCERACF
*          PROFILE:   <system id>.CURRENT
*                     <system id>.V001
*                       . . . . . .
*                     <system id>.Vnnn
*
*          INSTDATA field pseudo DSECT:
*
*            SECKEY   DS XL8        Secret key
*            KEYVER   DS CL3        Key version
*
*
*       User Profiles:
*       -------------
*
*          CLASS:     USER
*
*          USRDATA field pseudo DSECT:
*
*            USRDATA  DS OX                 User data field
*            DCEEYE   DC CL4'DCE('          Eye catcher
*            PRNOFF   DS AL1(PRN-USRDATA)   Offset to principal
*            PWDOFF   DS AL1(PWD-USRDATA)   Offset to encrypted
*            *                              DCE password
*            VEROFF   DS AL1(VER-USRDATA)   Offset to key version
```

Figure 17 (Part 3 of 39). Assembler Routine, USERSVC

```
*            PRN      DS OC                Principal name
*            *                             (null delimited)
*            PWD      DS OC                Encrypted DCE password
*            *                             (null delimited)
*            VER      DS CL3               Key version
*            DCEEND   DC CL1')'            end of field
*
* EXIT:
*
*    Request = 1    Logon to RACF
*
*        R15 = 0    RACF logon OK, ACEE created
*              4    RACF userid has been revoked
*              8    RACF userid not known to RACF or invalid passticket
*              12   DCE client is not using an authenicated RPC
*              16   DCE client does not have a UUID cross reference
*                   profile with RACF userid.
*              20   Requested function not recognized
*
*    Request = 2    Check authorisation to a resource
*
*        R15 = 0    access granted
*              8    access denied
*              12   resource is not known to RACF
*              16   class is not known to RACF
*              20   Requested function not recognized
*
*    Request = 3    Logoff RACF
*
*        R15 = 0    logged off RACF
*              8    userid not logged on
*              20   Requested function not recognized
*
*    Request = 4    Set encrypted DCE password
*
*        R15 = 0    Normal
*              8    RACF user profile USRDATA field does not contain
*                   DCE principal name, DCE password, and key version
*                   fields or the DCE fields are not valid.
*              12   RACF '<sysid>.CURRENT' profile not found
*              16   Unable to encrypt pad
*              20   Requested function not recognized
*              28   Unable to update user profile
*
*    Request = 5    Get DCE password
*
*        R15 = 0    normal
*              8    RACF user profile USRDATA field does not contain
*                   DCE principal name, DCE password, and key version
*                   fields or the DCE fields are not valid.
*              12   RACF '<sysid>.Vnnn' profile not found
*              16   Unable to encrypt the pad
*              20   Requested function not recognized
*
*    Request = 6    Extract DCE principal from USRDATA
*
*        R15 = 0    Normal completion
*              8    DCE keyword not found in ACEEINST
*             12    DCE field in ACEEINST is invalid
*             20    Requested function not recognized
*
*    Request = 7    set DCE principal in USRDATA
*
*        R15 = 0    Normal completion
*              8    DCE keyword not found in USRDATA
*             12    DCE field in USRDATA is invalid
*             16    Caller is not authorised to this request
*             20    Requested function not recognized
*
*    Request = 8    list USRDATA
```

Figure 17 (Part 4 of 39). Assembler Routine, USERSVC

```
*
*        R15 = 0    Normal completion
*              12
*              16    Caller is not authorised to this request
*              20    Requested function not recognized
*
*    Request = 9    get RACF sysid
*
*        R15 = 0    Normal completion
*              20    Requested function not recognized
*
*    Request = 10   set DCE uuid profile
*
*        R15 = 0    Normal completion
*               8    Unable to create/replace UUID profile
*              16    not authorised
*              20    Requested function not recognized
*
*    Request = 11   get DCE uuid profile
*
*        R15 = 0    Normal completion
*               8    Unable to find UUID profile
*              20    Requested function not recognized
*
* REGISTER USAGE
*
*        R0   ->  CENV, caller's R12 and r13 (on entry)
*        R1   ->  PARAMETER LIST
*        R2   -   WORK
*        R3   =   A(CVT) (on entry), then WORK
*        R4   =   A(TCB) (on entry), then WORK
*        R5   =   A(SVRB) (on entry), then WORK
*        R6   =   A(ENTRY POINT)  (on entry), then WORK
*        R7   =   A(ASCB) (on entry), then WORK
*        R8   ->  Constants area
*        R9   ->  Dynamic storage area
*        R10  ->  Dynamic Storage Area
*        R11  ->  BASE
*        R12  ->  RESERVED for 'C' environment
*        R13  ->  SAVE AREA / 'C' DSA
*        R14  ->  RETURN
*        R15 -    unchanged from caller
*
*
* NOTES
*
*     This module uses structured assembly macros, These macros
*     are copied from member STRMACS. Member STRREF in the same
*     library is a reference manual for the use of these macros.
*     The reference manual is marked up for use with BookMaster
*     but is readable in its unscripted form.
*
* CHANGE ACTIVITY:
*
*     Created 7/11/94 Eric Finkelstein ISSC Australia
*
*--------------------------------------------------------------------*
*
 PRINT GEN
 COPY STRMACS                          STRUCTURED MACROS
 COPY USVCNUM                          DCE/RACF USER SVC DETAILS
*
*--------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - MAIN '
*--------------------------------------------------------------------*
&USERSVCM START 0                      USER SVC CSECT
&USERSVCM AMODE 31                     ADDRESSING MODE
&USERSVCM RMODE ANY                    RESIDENCY MODE
*
*                                *----------------------------------*
```

Figure 17 (Part 5 of 39). Assembler Routine, USERSVC

```
*                                SAVE REGISTERS
*                               *---------------------------------*
*
 SAVE (14,12)                         SAVE REGISTERS IN HSA
*
*                               *---------------------------------*
*                                ESTABLISH ADDRESSING
*                               *---------------------------------*
*
 USING &USERSVCM,11                   ADDRESSING
 USING DSA,10,9                       ADDRESSING
 USING CONSTANT,8                     ADDRESSING
 LR 11,6                              LOAD BASE REGISTER
 LR 12,0                              R12 -> 'C' ENVIRONMENT
 CNOP 0,4                             FORCE ALIGNMENT TO WORD
 L 8,*+8                              R9 -> CONSTANTS
 B *+8                                BRANCH AROUND A(CONSTANT)
 DC A(CONSTANT)                       ADDRESS OF CONSTANT AREA
*
*                               *---------------------------------*
*                                GET DYNAMIC STORAGE AREA
*                               *---------------------------------*
*
 LR 12,0                              SAVE A(SVC PARAMETER LIST) IN 12
 LR 5,1                               SAVE A(PARAMETER LIST) IN R5
 L 2,=A(DSAE-DSA)                     R2 = L(DSA)
 GETMAIN R,LV=(2)                     GET DSA
 LR 10,1                              R10 -> DSA
 LA 9,2048(10)                        R9 = A(DSA + 2048)
 LA 9,2048(9)                         R9 = A(DSA + 4096)
*
*                               *---------------------------------*
*                                CLEAR DSA
*                               *---------------------------------*
*
 LR 2,10                              R2 -> DSA
 L 3,=A(DSAE-DSA)                     R3 =  L(DSA)
 XR 6,6                               R6 =  ZERO
 XR 7,7                               R7 =  ZERO
 MVCL 2,6                             INITITALISE TO NULLS
*
*                               *---------------------------------*
*                                CHAIN SAVE AREAS
*                               *---------------------------------*
*
 MVC SAVEDHSA(72),0(13)               SAVE ORIGINAL HSA
 ST 10,8(13)                          LSA IN HSA
 ST 13,4(10)                          HSA IN LSA
 LR 13,10                             LSA IN 13
*
*                               *---------------------------------*
*                                SAVE SELECTED ENTRY REGISTERS
*                               *---------------------------------*
*
 ST 4,PTRTCB                          SAVE A(TCB)
 ST 5,PTRPLIST                        SAVE A(PARAMETER LIST)
 MVC CENV(8),0(12)                    SAVE C R12 AND C R13
 MVI DBG,C'N'                         TURN ON/OFF DEBUG MESSAGES
*
*                               *---------------------------------*
*                                DETERMINE FUNCTION REQUESTED
*                               *---------------------------------*
*
 MVC RC,=F'0'                         INITIALISE RETURN CODE
 L 2,PTRPLIST                         R2 -> PARAMETER LIST
 L 2,0(2)                             R2 -> REQUEST
 L 2,0(2)                             R2 = REQUEST
 IF (C,2,LT,=F'1'),OR,(C,2,GT,=F'11')  IF REQUEST IS OUT OF RANGE
   MVC RC,=F'20'                      RETURN WITH ERROR
 ELSE
```

Figure 17 (Part 6 of 39). Assembler Routine, USERSVC

```
   CASENTRY 2                          SELECT USING CONTENTS OF R2
*
   CASE 1                              CASE RACF_logon
     CALL LOGON                        CALL LOGON SUBROUTINE
   CASE 2                              CASE RACF_check_authorisation
     CALL CHKAUTH                      CALL CHKAUTH SUBROUTINE
   CASE 3                              CASE RACF_logoff
     CALL LOGOFF                       CALL LOGOFF SUBROUTINE
   CASE 4                              CASE RACF_set_DCE_password
     CALL SETPWD                       CALL SETPWD SUBROUTINE
   CASE 5                              CASE RACF_get_DCE_password
     CALL GETPWD                       CALL GETPWD SUBROUTINE
   CASE 6                              CASE RACF_get_DCE_principal
     CALL GETPRIN                      CALL GETPRIN SUBROUTINE
   CASE 7                              CASE RACF_set_DCE_principal
     CALL SETPRIN                      CALL SETPRIN SUBROUTINE
   CASE 8                              CASE RACF_get_DCE_usrdata
     CALL GETUSR                       CALL GETUSR SUBROUTINE
   CASE 9                              CASE RACF_get_MVS_sysid
     CALL GETSYSID                     CALL GETSYSID SUBROUTINE
   CASE 10                             CASE RACF_set_DCE_uuid
     CALL SETUUID                      CALL SETUUID SUBROUTINE
   CASE 11                             CASE RACF_get_DCE_uuid
     CALL GETUUID                      CALL GETUUID SUBROUTINE
   ENDCASE                             END CASES
 ENDIF
*
*                                   *-----------------------------------*
*                                    RETURN
*                                   *-----------------------------------*
*
 L 7,RC                                  R7 = RC
 L 13,4(13)                              R13 -> HSA
 MVC 0(72,13),SAVEDHSA                   RESTORE HSA
 L 3,=A(DSAE-DSA)                        R3 = L(DSA)
 FREEMAIN R,LV=(3),A=(10)                FREE DSA
 LR 15,7                                 R15 = RC
 RETURN (14,12),RC=(15)                  RETURN
*
*
*-------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - LOGON SUBROUTINE'
*-------------------------------------------------------------------*
*
LOGON  DS 0H                             LOGON SUBROUTINE
 ENTRY LOGON
*
*                                   *-----------------------------------*
*                                    SAVE REGISTERS AND CHAIN SAVE AREAS
*                                   *-----------------------------------*
*
 SAVE (14,12)                            SAVE REGISTERS
 LA 2,LSA1                               R2 -> LSA 1
 ST 2,8(13)                              LSA IN HSA
 ST 13,4(2)                              HSA IN LSA
 LR 13,2                                 LSA IN 13
 LR 11,15                                LOAD BASE
 USING LOGON,11                          ESTABLISH ADDRESSING
 MVC RC,=F'0'                            RETURN CODE = 0
*
*                                   *-----------------------------------*
*                                    LOGOFF ANYONE ON THIS TCB ACEE
*                                   *-----------------------------------*
*
 L 2,PTRTCB                              R2 -> SAVED TCB
 L 2,TCBSENV-TCB(2)                      R2 -> ACEE
*
*                                   *-----------------------------------*
*                                    IS THERE AN ACEE POINTED BY TCBSENV
*                                   *-----------------------------------*
```

Figure 17 (Part 7 of 39). Assembler Routine, USERSVC

```
*
 IF    (C,2,NE,=F'0')                 IF ACEE OFF THE TCB
*
*                                *-----------------------------------*
*                                 ISSUE RACROUTE DELETE
*                                *-----------------------------------*
*
   MVC RAC02(RAC02E-RAC02C),RAC02C     COPY RACROUTE PARAMETER LIST
   RACROUTE REQUEST=VERIFY,                                          X
               ENVIR=DELETE,                                         X
               WORKA=RACWORKA,                                       X
               SYSTEM=NO,                                            X
               RELEASE=1.9.2,                                        X
               MF=(E,RAC02)
   ST 15,RC                            SAVE RETURN CODE
   IF (LTR,15,15,NZ)
     ST 15,DBGR15
     LA 1,RAC02
     MVC DBGMSG,=CL40'LOGON - LOGOFF'
     CALL DBGMSG2
   ENDIF
 ENDIF
*
*                                *-----------------------------------*
*                                 CALL GETPAC ('C' ROUTINE)
*                                *-----------------------------------*
*
 L 2,PTRPLIST                         R2 -> PARAMETER LIST
 L 3,4(2)                             R3 -> BINDING HANDLE
 ST 3,PLIST                           STORE IN GETPAC PARAMETER LIST
 LA 3,UUIDSTR                         R3 -> UUIDSTR
 ST 3,PLIST+4                         STORE IN GETPAC PARAMETER LIST
 OI PLIST+4,X'80'                     SET END OF LIST
 LA 1,PLIST                           R1 -> PLIST
*
 L 12,CENVR12                         R12 -> C ENVIRONMENT
 L 13,CENVR13                         R13 -> C ENVIRONMENT DSA
*
 CALL GETPAC                          CALL C SUBROUTINE GETPAC
 LA 13,LSA1                           RESTORE R13
*
 ST 15,RC                             SAVE RETURN CODE
*
 CALL DBGMSG1                         WRITE DEBUG MESSAGE 1
*
 IF (CLC,RC,EQ,=F'0')                 IF PACUUID OK
*
*                                *-----------------------------------*
*                                 GET USERID FROM UUID PROFILE
*                                *-----------------------------------*
*
   MVC SAVPLIST,PTRPLIST              SAVE CALLER'S PLIST POINTER
*
   LA 1,UUIDSTR                       R1 -> PRINCIPAL IN ACEE
   ST 1,PLIST+4                       STORE IN PARAMETER LIST
   LA 1,DCERACU                       R1 -> USERID
   ST 1,PLIST+8                       STORE IN PARAMETER LIST
   OI PLIST+8,X'80'                   SET END OF LIST
   LA 1,PLIST                         R1 -> PARAMETER LIST
   ST 1,PTRPLIST                      STORE OVER CALLER'S PLIST PTR
   CALL GETUUID                       GET USERID FROM UUID PROFILE
*
   MVC PTRPLIST,SAVPLIST              RESTORE CALLER'S PLIST POINTER
   IF (CLC,RC,EQ,=F'0')               IF RC FROM GETUUID = 0
*
*                                *-----------------------------------*
*                                 COPY IT TO USERID
*                                *-----------------------------------*
*
     XR 1,1                           R1 = 0 (COUNT)
```

Figure 17 (Part 8 of 39). Assembler Routine, USERSVC

```
     LA 2,DCERACU                     R2 -> DCE RACF USERID
     LA 3,USERID+1                    R3 -> USERID+1
     DO WHILE=(CLI,0(2),NE,X'00')     WHILE NOT NULL
       MVC 0(1,3),0(2)                COPY CHARACTER
       LA 1,1(1)                      BUMP COUNT
       LA 2,1(2)                      BUMP IN
       LA 3,1(3)                      BUMP OUT
     ENDDO
     STC 1,USERID                     STORE THE LENGTH IN USERID
*
*                               *----------------------------------*
*                                IF PASSTICKET NOT PRESENT
*                               *----------------------------------*
*
     L 2,PTRPLIST                     R2 -> CALLER'S PARAMETER LIST
     IF (TM,4(2),X'80',O)             IF HI BIT IS ON
*
*                               *----------------------------------*
*                                ISSUE RACROUTE - NO PASSTICKET
*                               *----------------------------------*
*
       MVC RACO1(RACO1E-RACO1C),RACO1C COPY RACROUTE PARAMETER LIST
       RACROUTE REQUEST=VERIFY,                                      X
               ENVIR=CREATE,                                         X
               USERID=USERID,                                        X
               APPL=APPLDCE,                                         X
               PASSCHK=NO,                                           X
               SYSTEM=NO,                                            X
               WORKA=RACWORKA,                                       X
               RELEASE=1.9.2,                                        X
               MF=(E,RACO1)
       ST 15,RC                        SAVE RETURN CODE
       IF (LTR,15,15,NZ)               IF RACROUTE OK
         ST 15,DBGR15                  SAVE RETURN CODE
         LA 1,RACO1
         MVC DBGMSG,=CL40'LOGON'
         CALL DBGMSG2
       ENDIF
*
     ELSE
*
*                               *----------------------------------*
*                                COPY PASSTICKET
*                               *----------------------------------*
*
       XR 1,1                         R1 = 0 (COUNT)
       L 2,PTRPLIST                   R2 -> CALLER'S PARAMETER LIST
       L 2,8(2)                       R2 -> CALLER'S PASSTICKET
       LA 3,PASSTICK+1                R3 -> PASSTICK+1
       DO WHILE=(CLI,0(2),NE,X'00')   WHILE NOT NULL
         MVC 0(1,3),0(2)              COPY CHARACTER
         LA 1,1(1)                    BUMP COUNT
         LA 2,1(2)                    BUMP IN
         LA 3,1(3)                    BUMP OUT
       ENDDO
       STC 1,PASSTICK                 STORE THE LENGTH IN PASSTICK
*
*                               *----------------------------------*
*                                BUILD APPL NAME FROM SMCASID
*                               *----------------------------------*
*
       L 2,16                         R2 -> CVT
       L 2,CVTSMCA-CVT(2)             R2 -> SMCA
       LA 2,SMCASID-SMCABASE(2)       R2 -> SMF SYSTEM ID
       MVC APPLPTKT,=CL8'DCE'         INITILISE APPLPTKT
       LA 3,APPLPTKT+3                R3 -> SID IN APPNAME
       L  1,=F'4'                     R1 = LOOP COUNT
       DO WHILE=(LTR,1,1,NZ)
         IF (CLI,0(2),GE,C'A')        IF  NOT A SPECIAL CHARACTER
           MVC 0(1,3),0(2)            COPY CHARACTER
```

Figure 17 (Part 9 of 39). Assembler Routine, USERSVC

```
          LA 3,1(3)                    BUMP OUTPUT
        ENDIF
        S 1,=F'1'                      DECREMENT COUNT
        LA 2,1(2)                      BUMP INPUT
      ENDDO
*
      IF (CLI,DBG,EQ,C'Y')
        MVC WTO(WTOE-WTOC),WTOC        COPY WTO LIST TO DSA
        MVC WTO+4(7),=C'TICKET='
*
        LA 3,WTO+11
        LA 2,PASSTICK+1                R2 -> TICKET
        MVC 0(8,3),0(2)               COPY CHARACTER
*
        LA 3,WTO+20
        MVC 0(9,3),=C'APPLNAME='
*
        LA 3,WTO+29
        LA 2,APPLPTKT                  R2 -> APPLNAME
        MVC 0(8,3),0(2)                COPY CHARACTER
        WTO MF=(E,WTO)
      ENDIF
*
*
*                               *----------------------------------*
*                                ISSUE RACROUTE - PASSTICKET
*                               *----------------------------------*
*
      MVC RAC09(RAC09E-RAC09C),RAC09C COPY RACROUTE PARAMETER LIST
      RACROUTE REQUEST=VERIFY,                                      X
            ENVIR=CREATE,                                           X
            USERID=USERID,                                          X
            APPL=APPLPTKT,                                          X
            PASSCHK=YES,                                            X
            PASSWRD=PASSTICK,                                       X
            SYSTEM=NO,                                              X
            WORKA=RACWORKA,                                         X
            RELEASE=1.9.2,                                          X
            MF=(E,RAC09)
      ST 15,RC                          SAVE RETURN CODE
      IF (LTR,15,15,NZ)                 IF RACROUTE OK
        ST 15,DBGR15                    SAVE RETURN CODE
        LA 1,RAC09
        MVC DBGMSG,=CL40'LOGON PASSTICKET'
        CALL DBGMSG2
      ENDIF
    ENDIF
  ENDIF
ENDIF
*
*                               *----------------------------------*
*                                RETURN
*                               *----------------------------------*
*
 L     13,4(13)                        R13 -> HSA
RETURN (14,12)                         RETURN
*
*--------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - CHKAUTH SUBROUTINE'
*--------------------------------------------------------------------*
*
CHKAUTH DS 0H                          CHKAUTH SUBROUTINE
 ENTRY CHKAUTH
*
*                               *----------------------------------*
*                                SAVE REGISTERS AND CHAIN SAVE AREAS
*                               *----------------------------------*
*
 SAVE (14,12)                          SAVE REGISTERS
 LA 2,LSA2                             R2 -> LSA 2
```

Figure 17 (Part 10 of 39). Assembler Routine, USERSVC

```
 ST 2,8(13)                          LSA IN HSA
 ST 13,4(2)                          HSA IN LSA
 LR 13,2                             LSA IN 13
 LR 11,15                            LOAD BASE
 USING CHKAUTH,11                    ESTABLISH ADDRESSING
 MVC RC,=F'0'                        RETURN CODE = 0
*
*                                  *----------------------------------*
*                                   GET CLASS PARAMETER
*                                  *----------------------------------*
*
 MVC CLASS,=CL256' '                 COPY BLANKS
 XR 1,1                              R1 = 0 (COUNT)
 L 2,PTRPLIST                        R2 -> PARAMETER LIST
 L 3,8(2)                            R3 -> CLASS STRING
 LA 4,CLASS+1                        R4 -> CLASS
 DO WHILE=(CLI,0(3),NE,X'00')        DO WHILE NOT NULL TERMINATOR
   MVC 0(1,4),0(3)                   COPY CHARACTER
   OI 0(4),X'40'                     MAKE UPPERCASE
   LA 1,1(1)                         BUMP R1
   LA 3,1(3)                         BUMP R3
   LA 4,1(4)                         BUMP R4
 ENDDO
 STC 1,CLASS                         STORE LENGTH
*
*                                  *----------------------------------*
*                                   GET ENTITY PARAMETER
*                                  *----------------------------------*
*
 L 2,PTRPLIST                        R2 -> PARAMETER LIST
 L 3,12(2)                           R3 -> ENTITY STRING
 LA 4,ENTITY                         R4 -> ENTITY
 MVC ENTITY,=CL256' '                COPY BLANKS
 DO WHILE=(CLI,0(3),NE,X'00')        DO WHILE NOT NULL TERMINATOR
   MVC 0(1,4),0(3)                   COPY CHARACTER
   OI 0(4),X'40'                     MAKE UPPERCASE
   LA 3,1(3)                         BUMP R3
   LA 4,1(4)                         BUMP R4
 ENDDO
*
*                                  *----------------------------------*
*                                   GET ACCESS PARAMETER
*                                  *----------------------------------*
*
 L 2,PTRPLIST                        R2 -> PARAMETER LIST
 L 3,16(2)                           R3 -> ENTITY STRING
 MVI ACCESS,X'02'                    DEFAULT IS READ
*
*                                    IF READ REQUESTED
*
 IF (CLI,0(3),EQ,C'R'),OR,(CLI,0(3),EQ,C'r')
   MVI ACCESS,X'02'                  ACCESS = READ
 ENDIF
*
*                                    IF UPDATE REQUESTED
*
 IF (CLI,0(3),EQ,C'U'),OR,(CLI,0(3),EQ,C'u')
   MVI ACCESS,X'04'                  ACCESS = UPDATE
 ENDIF
*
*                                    IF CONTROL REQUESTED
*
 IF (CLI,0(3),EQ,C'C'),OR,(CLI,0(3),EQ,C'c')
   MVI ACCESS,X'08'                  ACCESS = CONTROL
 ENDIF
*
*                                    IF ALTER REQUESTED
*
 IF (CLI,0(3),EQ,C'A'),OR,(CLI,0(3),EQ,C'a')
   MVI ACCESS,X'80'                  ACCESS = ALTER
```

Figure 17 (Part 11 of 39). Assembler Routine, USERSVC

```
 ENDIF
*
*                               *-----------------------------------*
*                                ISSUE RACROUTE
*                               *-----------------------------------*
*
 XR 2,2                              R2 = 0
 IC 2,ACCESS                         R2 = ACCESS
 MVC RAC03(RAC03E-RAC03C),RAC03C     COPY RACROUTE PARAMETER LIST
 RACROUTE REQUEST=AUTH,                                          X
               WORKA=RACWORKA,                                   X
               CLASS=CLASS,                                      X
               ENTITY=(ENTITY),                                  X
               ATTR=(2),                                         X
               RELEASE=1.9.2,                                    X
               MF=(E,RAC03)
 IF (LTR,15,15,NZ)
   ST 15,RC
   ST 15,DBGR15
   LA 1,RAC03
   MVC DBGMSG,=CL40'CHKAUTH'
   CALL DBGMSG2
 ENDIF
*
*                               *-----------------------------------*
*                                RETURN
*                               *-----------------------------------*
*
 L     13,4(13)                      R13 -> HSA
 RETURN (14,12)                      RETURN
*
*--------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - LOGOFF SUBROUTINE'
*--------------------------------------------------------------------*
*
LOGOFF DS OH                         LOGOFF SUBROUTINE
 ENTRY LOGOFF
*
*                               *-----------------------------------*
*                                SAVE REGISTERS AND CHAIN SAVE AREAS
*                               *-----------------------------------*
*
 SAVE (14,12)                        SAVE REGISTERS
 LA 2,LSA3                           R2 -> LSA 3
 ST 2,8(13)                          LSA IN HSA
 ST 13,4(2)                          HSA IN LSA
 LR 13,2                             LSA IN 13
 LR 11,15                            LOAD BASE
 USING LOGOFF,11                     ESTABLISH ADDRESSING
 MVC RC,=F'0'                        RETURN CODE = 0
*
*                               *-----------------------------------*
*                                FIND TCB ACEE
*                               *-----------------------------------*
*
 L 2,PTRTCB                          R2 -> SAVED TCB
 L 2,TCBSENV-TCB(2)                  R2 -> ACEE
*
*                               *-----------------------------------*
*                                IS THERE AN ACEE POINTED BY TCBSENV
*                               *-----------------------------------*
*
 IF    (C,2,EQ,=F'0')                IF NO ACEE OFF THE TCB
   MVC RC,=F'8'                      RETURN WITH ERROR
 ELSE
*
*                               *-----------------------------------*
*                                ISSUE RACROUTE DELETE
*                               *-----------------------------------*
*
```

Figure 17 (Part 12 of 39). Assembler Routine, USERSVC

```
   MVC RAC02(RAC02E-RAC02C),RAC02C     COPY RACROUTE PARAMETER LIST
   RACROUTE REQUEST=VERIFY,                                        X
               ENVIR=DELETE,                                       X
               WORKA=RACWORKA,                                     X
               SYSTEM=NO,                                          X
               RELEASE=1.9.2,                                      X
               MF=(E,RAC02)
   ST 15,RC                            SAVE RETURN CODE
   IF (LTR,15,15,NZ)
     ST 15,DBGR15
     LA 1,RAC02
     MVC DBGMSG,=CL40'LOGOFF'
     CALL DBGMSG2
   ENDIF
 ENDIF
*
*                                *----------------------------------*
*                                 RETURN
*                                *----------------------------------*
*
 L     13,4(13)                       R13 -> HSA
 RETURN (14,12)                       RETURN
*
*--------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - SETPWD SUBROUTINE'
*--------------------------------------------------------------------*
*
SETPWD DS OH                            SETPWD SUBROUTINE
 ENTRY SETPWD
*
*                                *----------------------------------*
*                                 SAVE REGISTERS AND CHAIN SAVE AREAS
*                                *----------------------------------*
*
 SAVE (14,12)                         SAVE REGISTERS
 LA 2,LSA4                            R2 -> LSA 4
 ST 2,8(13)                           LSA IN HSA
 ST 13,4(2)                           HSA IN LSA
 LR 13,2                              LSA IN 13
 LR 11,15                             LOAD BASE
 USING SETPWD,11                      ESTABLISH ADDRESSING
 MVC RC,=F'0'                         RETURN CODE = 0
*
*                                *----------------------------------*
*                                 FIND CURRENT TCB
*                                *----------------------------------*
*
 L 2,16                               R2 -> CVT
 L 2,CVTTCBP-CVTMAP(2)                R2 -> A(NEXT TCB, CURRENT TCB)
 L 2,4(2)                             R2 -> CURRENT TCB
*
*                                *----------------------------------*
*                                 FIND ACEE (TCB OR ASXB)
*                                *----------------------------------*
*
 L 2,TCBSENV-TCB(2)                   R2 -> ACEE
 IF (C,2,EQ,=F'0')                    IS THERE NO ACEE IN TCBSENV
   L 2,16                             R2 -> CVT
   L 2,CVTTCBP-CVTMAP(2)              R2 -> A(NEXT TCB, CURRENT TCB)
   L 2,12(2)                          R2 -> CURRENT ASCB
   L 2,ASCBASXB-ASCB(2)               R2 -> ASXB
   L 2,ASXBSENV-ASXB(2)               R2 -> ACEE
 ENDIF
 ST 2,PTRACEE                         SAVE A(ACEE)
*
*                                *----------------------------------*
*                                 GET RACF USERID FROM ACEE
*                                *----------------------------------*
*
 MVC USERID,ACEEUSER-ACEE(2)          COPY USERID FROM ACEE
```

Figure 17 (Part 13 of 39). Assembler Routine, USERSVC

```
*
*                               *---------------------------------*
*                                GET DCE PRINCIPAL NAME FROM USRDATA
*                               *---------------------------------*
*
 MVC SAVPLIST,PTRPLIST                SAVE CALLER'S PLIST POINTER
 LA 1,USRPRIN                         R1 -> PRINCIPAL IN ACEE
 ST 1,PLIST+4                         STORE IN PARAMETER LIST
 LA 1,PLIST                           R1 -> PARAMETER LIST
 ST 1,PTRPLIST                        STORE OVER CALLER'S PLIST PTR
 CALL GETPRIN                         GET PRINCIPAL FROM ACEE
 MVC PTRPLIST,SAVPLIST                RESTORE CALLER'S PLIST POINTER
*
 IF (CLC,RC,EQ,=F'0')                 IF PRINCIPAL FOUND
*
*                               *---------------------------------*
*                                EXTRACT SECRET KEY FROM;
*                                <SYSID>.CURRENT
*                               *---------------------------------*
*
   MVC CLASS,=CL8'$DCERACF'           CLASS = $DCERACF
*
*                                     BUILD PROFILE NAME
*
   MVC ENTITY,=CL256' '               BLANK ENTITY FIELD
   L 2,16                             R2 -> CVT
   MVC ENTITY(8),CVTSNAME-CVTMAP(2)   COPY SYSTEM NAME TO ENTITY
   LA 2,ENTITY                        R2 -> ENTITY FIELD
   DO WHILE=(CLI,0(2),NE,C' ')        FIND FIRST BLANK
     LA 2,1(2)                        BUMP R2
   ENDDO
   MVC 0(8,2),=C'.CURRENT'            MOVE SUFFIX TO ENTITY
*
*                                     ISSUE RACROUTE EXTRACT
*
   MVC RAC04(RAC04E-RAC04C),RAC04C    COPY RACROUTE PARAMETERS
   RACROUTE REQUEST=EXTRACT,                                       X
               TYPE=EXTRACT,                                       X
               CLASS=CLASS,                                        X
               ENTITY=ENTITY,                                      X
               FIELDS=INSTDATA,                                    X
               SEGMENT=BASE,                                       X
               WORKA=RACWORKA,                                     X
               RELEASE=1.9.2,                                      X
               MF=(E,RAC04)
   LR  2,1                            SAVE A(RESULT)
   ST  2,SAVEBUF                      SAVE A(EXTRACT BUFFER)
*
*                                     TEST RESULT
*
   IF (LTR,15,15,NZ)                  IF AN ERROR
     MVC RC,=F'12'                    SET RETURN CODE
     ST 15,DBGR15
     LA 1,RAC04
     MVC DBGMSG,=CL40'SETPWD - CURRENT KEY'
     CALL DBGMSG2
   ELSE
*
*                                     COPY BUFFER
*
     LR 4,2                           R4 -> EXTRACT RESULT
     AH 4,EXTWOFF-EXTWKEA(2)          R4 = R4 + OFFSET TO INSTDATA
     L 5,0(4)                         R5 = L(INSTDATA)
     LA 5,4(5)                        R5 = L(INSTDATA) + LENGTH FIELD
     LA 6,KEYLEN                      R6 -> KEYLEN
     LR 7,5                           R7 = LENGTH TO MOVE
     MVCL 6,4                         COPY RESULTS
*
*                                     FREE BUFFER
*
```

Figure 17 (Part 14 of 39). Assembler Routine, USERSVC

```
     L 2,SAVEBUF                       R2 -> EXTRACT BUFFER
     XR 3,3                            ZERO OUT R3
     XR 4,4                            ZERO OUT R4
     USING EXTWKEA,2                   BASE THE AREA ON R2
     ICM 3,1,EXTWSP                    MOVE THE AREA SUBPOOL INTO R3
     ICM 4,7,EXTWLN                    MOVE THE AREA LENGTH INTO R4
     DROP 2                            DROP BASING ON REGISTER 2
     FREEMAIN R,LV=(4),A=(2),SP=(3)    FREE EXTRACT BUFFER
*
*                                *-----------------------------------*
*                                 ENCRYPT USERID UNDER SYSTEM SECRET
*                                 KEY AS THE PAD
*                                *-----------------------------------*
*
     MVC PAD,KEY                       COPY SECRET KEY TO PAD
     MVI PADLEN,X'08'                  INSERT LENGTH OF PAD
*
*                                      ISSUE RACROUTE ENCRYPT
*
     MVC RAC05(RAC05E-RAC05C),RAC05C   COPY MACRO TO WORK AREA
     RACROUTE REQUEST=EXTRACT,                                          X
               TYPE=ENCRYPT,                                            X
               ENTITY=USERID,                                           X
               ENCRYPT=(PADAREA,STDDES),                                X
               WORKA=RACWORKA,                                          X
               RELEASE=1.9.2,                                           X
               MF=(E,RAC05)
*
*                                      TEST RESULT
*
     IF (LTR,15,15,NZ)                 IF AN ERROR
       MVC RC,=F'16'                   SET RETURN CODE
       ST 15,DBGR15
       LA 1,RAC05
       MVC DBGMSG,=CL40'SETPWD - USERID '
       CALL DBGMSG2
     ELSE
*
*                                *-----------------------------------*
*                                 BUILD USRDATA
*                                *-----------------------------------*
*
       XR 1,1                          R1 = 0 (OFFSET)
       MVC USRDATA(4),=C'DCE('         MOVE KEYWORD
       LA 1,4(1)                       BUMP OFFSET
       MVC USRDATA+4(3),=X'070000'     INITIALISE OFFSETS
       LA 1,3(1)                       BUMP OFFSET
*
*                                      COPY PRINCIPAL TO USRDATA
*
       LA 2,USRPRIN                    R2 -> PRINCIPAL NAME
       LA 3,USRDATA(1)                 R3 -> OUTPUT
       DO WHILE=(CLI,0(2),NE,X'00')    DO WHILE NOT NULL
         MVC 0(1,3),0(2)               MOVE CHARACTER
         LA 1,1(1)                     BUMP R1 (OFFSET)
         LA 2,1(2)                     BUMP R2 (INPUT)
         LA 3,1(3)                     BUMP R2 (OUTPUT)
       ENDDO
       MVI 0(3),X'00'                  INSERT DELIMITER AFTER NAME
       LA 1,1(1)                       BUMP R1 (OFFSET)
       LA 3,1(3)                       BUMP R3 (OUTPUT)
*
*                                      COPY AND XOR PASSWORD WITH PAD
*
       STC 1,USRDATA+5                 STORE OFFSET TO PASSWORD
       L 2,PTRPLIST                    R2 -> PLIST
       L 2,4(2)                        R2 -> CLEAR PASSWORD
       LA 5,PAD+8                      R5 -> END OF PAD + 1
       LA 4,PAD                        R4 -> PAD
       DO WHILE=(CLI,0(2),NE,X'00')    DO WHILE NOT NULL
```

Figure 17 (Part 15 of 39). Assembler Routine, USERSVC

```
        MVC 0(1,3),0(2)               COPY CHARACTER
        XC 0(1,3),0(4)                XOR PAD CHARACTER OVER IT
        LA 1,1(1)                     BUMP R1 (OFFSET)
        LA 2,1(2)                     BUMP R2 (INPUT)
        LA 3,1(3)                     BUMP R3 (OUTPUT)
        LA 4,1(4)                     BUMP R4 (PAD)
        IF (CR,4,EQ,5)                IF END OF PAD
          LA 4,PAD                    R4 -> PAD
        ENDIF
      ENDDO
      MVI 0(3),X'00'                  INSERT DELIMITER
      LA 1,1(1)                       BUMP R1 (OFFSET)
      LA 3,1(3)                       BUMP R3
*
*                                     COPY KEY VERSION USED
*
      STC 1,USRDATA+6                 STORE OFFSET TO VERSION
      MVC 0(3,3),KEYVER               COPY KEY VERSION
      LA 3,3(3)                       BUMP R3
      MVI 0(3),C')'                   INSERT CLOSE BRACKET
      LA 3,1(3)                       BUMP R3
*
*                                     STORE LENGTH OF USRDATA
*
      LA 4,USRDATA                    R4 -> BEGINNING OF USRDATA
      SR 3,4                          R3 = LENGTH OF USRDATA
      ST 3,USRDATAL                   STORE LENGTH IN USRDATAL
*
*                                     DEBUG MESSAGE
*
      IF (CLI,DBG,EQ,C'Y')
*
        MVC WTO(WTOE-WTOC),WTOC       COPY WTO LIST TO DSA
        MVC WTO+4(11),=C'SETPWD PAD='
        L 1,=F'8'                     R1 = 8 (LENGTH OF INPUT)
        LA 2,PAD                      R2 -> PAD
        LA 3,WTO+15                   R3 -> MESSAGE (OUTPUT)
        CALL HEXCONV                  CONVERT TO HEX
        WTO MF=(E,WTO)
*
        MVC WTO(WTOE-WTOC),WTOC       COPY WTO LIST TO DSA
        MVC WTO+4(15),=C'SETPWD USRDATA='
        L 1,USRDATAL                  R1 = USRDATAL
        LA 2,USRDATA                  R2 -> USRDATA
        LA 3,WTO+19                   R3 -> MESSAGE (OUTPUT)
        CALL HEXCONV                  CONVERT TO HEX
        WTO MF=(E,WTO)
      ENDIF
*
*                              *-----------------------------------*
*                               UPDATE USRDATA IN RACF PROFILE
*                              *-----------------------------------*
*
      MVC CLASS,=CL8'USER'            CLASS = USER
      MVC RAC06(RAC06E-RAC06C),RAC06C COPY MACRO TO WORK AREA
      RACROUTE REQUEST=EXTRACT,                                       X
              TYPE=REPLACE,                                           X
              ENTITY=USERID+1,                                        X
              CLASS=CLASS,                                            X
              FIELDS=USRDATAC,                                        X
              SEGDATA=USRDATAL,                                       X
              SEGMENT=BASE,                                           X
              WORKA=RACWORKA,                                         X
              RELEASE=1.9.2,                                          X
              MF=(E,RAC06)
*
*                                     TEST RESULT
*
      IF (LTR,15,15,NZ)               IF AN ERROR
        MVC RC,=F'28'                 SET RETURN CODE
```

Figure 17 (Part 16 of 39). Assembler Routine, USERSVC

```
        ST 15,DBGR15
        LA 1,RAC06
        MVC DBGMSG,=CL40'SETPWD - UPDATE '
        CALL DBGMSG2
      ENDIF
    ENDIF
  ENDIF
 ENDIF
*
*                               *---------------------------------*
*                                RETURN
*                               *---------------------------------*
*
 L     13,4(13)                       R13 -> HSA
 RETURN (14,12)                       RETURN
*
*--------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - GETPWD SUBROUTINE'
*--------------------------------------------------------------------*
*
GETPWD DS OH                          GETPWD SUBROUTINE
 ENTRY GETPWD
*
*                               *---------------------------------*
*                                SAVE REGISTERS AND CHAIN SAVE AREAS
*                               *---------------------------------*
*
 SAVE (14,12)                         SAVE REGISTERS
 LA 2,LSA5                            R2 -> LSA 5
 ST 2,8(13)                           LSA IN HSA
 ST 13,4(2)                           HSA IN LSA
 LR 13,2                              LSA IN 13
 LR 11,15                             LOAD BASE
 USING GETPWD,11                      ESTABLISH ADDRESSING
 MVC RC,=F'0'                         RETURN CODE = 0
*
*                               *---------------------------------*
*                                FIND CURRENT TCB
*                               *---------------------------------*
*
 L 2,16                               R2 -> CVT
 L 2,CVTTCBP-CVTMAP(2)                R2 -> A(NEXT TCB, CURRENT TCB)
 L 2,4(2)                             R2 -> CURRENT TCB
*
*                               *---------------------------------*
*                                FIND ACEE (TCB OR ASXB)
*                               *---------------------------------*
*
 L 2,TCBSENV-TCB(2)                   R2 -> ACEE
 IF (C,2,EQ,=F'0')                    IF THERE NO ACEE IN TCBSENV
   L 2,16                             R2 -> CVT
   L 2,CVTTCBP-CVTMAP(2)              R2 -> A(NEXT TCB, CURRENT TCB)
   L 2,12(2)                          R2 -> CURRENT ASCB
   L 2,ASCBASXB-ASCB(2)               R2 -> ASXB
   L 2,ASXBSENV-ASXB(2)               R2 -> ACEE
 ENDIF
 ST 2,PTRACEE                         SAVE A(ACEE)
*
*                               *---------------------------------*
*                                GET RACF USERID FROM ACEE
*                               *---------------------------------*
*
 MVC USERID,ACEEUSER-ACEE(2)          COPY USERID FROM ACEE
*
*                               *---------------------------------*
*                                GET USRDATA
*                               *---------------------------------*
*
*
 MVC CLASS,=CL8'USER'
```

Figure 17 (Part 17 of 39). Assembler Routine, USERSVC

```
 MVC RAC04(RAC04E-RAC04C),RAC04C       COPY RACROUTE PARAMETERS
 RACROUTE REQUEST=EXTRACT,                                          X
               TYPE=EXTRACT,                                        X
               CLASS=CLASS,                                         X
               ENTITY=USERID+1,                                     X
               FIELDS=USRDATAC,                                     X
               SEGMENT=BASE,                                        X
               WORKA=RACWORKA,                                      X
               RELEASE=1.9.2,                                       X
               MF=(E,RAC04)
 LR 2,1                                SAVE A(RESULT)
 ST 2,SAVEBUF                          SAVE A(EXTRACT BUFFER)
*
*                                      TEST RESULT
*
 IF (LTR,15,15,NZ)                     IF AN ERROR
   MVC RC,=F'12'                       SET RETURN CODE
   ST 15,DBGR15
   LA 1,RAC04
   MVC DBGMSG,=CL40'GETPWD   - GET USRDATA'
   CALL DBGMSG2
 ELSE
*
*                                      COPY BUFFER
*
   LR 4,2                              R4 -> EXTRACT RESULT
   AH 4,EXTWOFF-EXTWKEA(2)             R4 = R4 + OFFSET TO USRDATA
   LA 4,4(4)                           R5 = L(FIRST USRDATA)
   L  5,0(4)                           R5 = L(FIRST USRDATA)
   LA 5,4(5)                           R5 = L(FIRST USRDATA) + LENGTH
   LA 6,USRDATAL                       R6 -> KEY
   LR 7,5                              R7 = LENGTH TO MOVE
   MVCL 6,4                            COPY RESULTS
*
*                                      FREE BUFFER
*
   L 2,SAVEBUF                         R2 -> EXTRACT BUFFER
   XR 3,3                              ZERO OUT R3
   XR 4,4                              ZERO OUT R4
   USING EXTWKEA,2                     BASE THE AREA ON R2
   ICM 3,1,EXTWSP                      MOVE THE AREA SUBPOOL INTO R3
   ICM 4,7,EXTWLN                      MOVE THE AREA LENGTH INTO R4
   DROP 2                              DROP BASING ON REGISTER 2
   FREEMAIN R,LV=(4),A=(2),SP=(3)      FREE EXTRACT BUFFER
*
*                                *-----------------------------------*
*                                 FIND DCE KEYWORD IN USRDATA
*                                *-----------------------------------*
*
   LA 3,USRDATA                        R3 -> USRDATA
   IF (CLC,USRDATAL,EQ,=F'0'),OR,      IF NO USRDATA                  X
               (CLC,0(4,3),NE,=C'DCE(')  OR NO DCE KEYWORD
     MVC RC,=F'8'                      RETURN WITH ERROR
   ELSE
*
*                                *-----------------------------------*
*                                 IS PASSWORD AND VERSION PRESENT
*                                *-----------------------------------*
*
     IF (CLI,USRDATA+5,EQ,X'00')       IF OFFSET TO PASSWORD = 0
       MVC RC,=F'8'                    INDICATE WRONG DCE DATA
     ELSE
       IF (CLI,USRDATA+6,EQ,X'00')     IF OFFSET TO VERSION = 0
         MVC RC,=F'8'                  INDICATE WRONG DCE DATA
       ELSE
*
*                                *-----------------------------------*
*                                 COPY ENCRYPTED PASSWORD TO PARAMETER
*                                *-----------------------------------*
*
```

Figure 17 (Part 18 of 39). Assembler Routine, USERSVC

```
         XR 1,1                        R1 = 0
         IC 1,USRDATA+5                R1 = OFFSET TO PASSWORD
         LA 2,USRDATA(1)               R2 -> ENCRYPTED PASSWORD
*
         L 3,PTRPLIST                  R3 -> CALLER'S PARAMETER LIST
         L 3,4(3)                      R3 -> CALLER'S PARAMETER
*
         XR 4,4                        R4 = 0
         IC 4,USRDATA+6                R4 = OFFSET TO VERSION
         LA 5,USRDATA(4)               R5 -> VERSION (END OF PASSWORD)
         SR 4,1                        R4 = LENGTH OF PASSWORD
         ST 4,PWDLEN                   PWDLEN = LENGTH (USED LATER)
*
         DO WHILE=(CR,2,NE,5)          DO WHILE NOT END OF STRING
           MVC 0(1,3),0(2)             COPY CHARACTER
           LA 2,1(2)                   BUMP R2
           LA 3,1(3)                   BUMP R3
         ENDDO
*
*                               *-----------------------------------*
*                                SAVE KEY VERSION FROM USRDATA
*                               *-----------------------------------*
*
         XR 1,1                             R1 = 0
         IC 1,USRDATA+6                     R1 = OFFSET TO VERSION
         LA 2,USRDATA(1)                    R2 -> VERSION
         IF (CLI,0(2),NE,C')')              IF PASSWORD IN PROFILE
           MVC KEYVERA,0(2)                 KEYVERA = VERSION FROM USRDATA
*
*                               *-----------------------------------*
*                                EXTRACT SECRET KEY FROM;
*                                <SYSID>.VNNN
*                               *-----------------------------------*
*
           MVC CLASS,=CL8'$DCERACF'
*
*                                       BUILD PROFILE NAME
*
           MVC ENTITY,=CL256' '          BLANK ENTITY FIELD
           L 2,16                        R2 -> CVT
           MVC ENTITY(8),CVTSNAME-CVTMAP(2) COPY SYSTEM NAME TO ENTITY
           LA 2,ENTITY                   R2 -> ENTITY FIELD
           DO WHILE=(CLI,0(2),NE,C' ')       FIND FIRST BLANK
             LA 2,1(2)                   BUMP R2
           ENDDO
           MVC 0(2,2),=C'.V'             MOVE SUFFIX TO ENTITY
           LA 2,2(2)                     BUMP R2
           MVC 0(3,2),KEYVERA            COPY KEY VERSION
*
*                                       ISSUE RACROUTE EXTRACT
*
           MVC RAC04(RAC04E-RAC04C),RAC04C COPY RACROUTE PARAMETERS
           RACROUTE REQUEST=EXTRACT,                                   X
               TYPE=EXTRACT,                                           X
               CLASS=CLASS,                                            X
               ENTITY=ENTITY,                                          X
               FIELDS=INSTDATA,                                        X
               SEGMENT=BASE,                                           X
               WORKA=RACWORKA,                                         X
               RELEASE=1.9.2,                                          X
               MF=(E,RAC04)
           LR 2,1                        SAVE A(RESULT)
           ST 2,SAVEBUF                  SAVE A(EXTRACT BUFFER)
*
*                                       TEST RESULT
*
           IF (LTR,15,15,NZ)             IF AN ERROR
             MVC RC,=F'12'               SET RETURN CODE
             ST 15,DBGR15
             LA 1,RAC04
```

Figure 17 (Part 19 of 39). Assembler Routine, USERSVC

```
            MVC DBGMSG,=CL40'GETPWD   - GET VNNN'
            CALL DBGMSG2
          ELSE
*
*                                 COPY BUFFER
*
            LR 4,2                R4 -> EXTRACT RESULT
            AH 4,EXTWOFF-EXTWKEA(2)  R4 = R4 + OFFSET TO INSTDATA
            L 5,0(4)              R5 = L(INSTDATA)
            LA 5,4(5)             R5 = L(INSTDATA) + LENGTH FIELD
            LA 6,KEYLEN           R6 -> KEYLEN
            LR 7,5                R7 = LENGTH TO MOVE
            MVCL 6,4              COPY RESULTS
*
*                                 FREE BUFFER
*
            L 2,SAVEBUF           R2 -> EXTRACT BUFFER
            XR 3,3                ZERO OUT R3
            XR 4,4                ZERO OUT R4
            USING EXTWKEA,2       BASE THE AREA ON R2
            ICM 3,1,EXTWSP        MOVE THE AREA SUBPOOL INTO R3
            ICM 4,7,EXTWLN        MOVE THE AREA LENGTH INTO R4
            DROP 2                DROP BASING ON REGISTER 2
            FREEMAIN R,LV=(4),A=(2),SP=(3) FREE EXTRACT BUFFER
*
*                              *---------------------------------*
*                               ENCRYPT USERID UNDER SYSTEM SECRET
*                               KEY AS THE PAD
*                              *---------------------------------*
*
            MVC PAD,KEY           COPY SECRET KEY TO PAD
            MVI PADLEN,X'08'      INSERT LENGTH OF PAD
*
*                                 ISSUE RACROUTE ENCRYPT
*
            MVC RAC05(RAC05E-RAC05C),RAC05C COPY MACRO TO WORK AREA
            RACROUTE REQUEST=EXTRACT,                              X
              TYPE=ENCRYPT,                                        X
              ENTITY=USERID,                                       X
              ENCRYPT=(PADAREA,STDDES),                            X
              WORKA=RACWORKA,                                      X
              RELEASE=1.9.2,                                       X
              MF=(E,RAC05)
*
*                                 TEST RESULT
*
            IF (LTR,15,15,NZ)     IF AN ERROR
              MVC RC,=F'16'       SET RETURN CODE
              ST 15,DBGR15
              LA 1,RAC04
              MVC DBGMSG,=CL40'GETPWD   - ENCRYPT USERID'
              CALL DBGMSG2
            ELSE
*
*                              *---------------------------------*
*                                XOR PAD ONTO THE ENCRYPTED PASSWORD
*                              *---------------------------------*
*
              L 2,PTRPLIST        R2 -> PLIST
              L 2,4(2)            R2 -> ENCRYPTED PASSWORD
              LA 1,PAD+8          R1 -> END OF PAD + 1
              LA 3,PAD            R3 -> PAD
              L 4,PWDLEN          R4 = SAVED PASSWORD LENGTH
              S 4,=F'1'           DON'T COUNT THE NULL DELIMITOR
              DO WHILE=(LTR,4,4,NZ)  DO WHILE NOT ZERO
                XC 0(1,2),0(3)    XOR PAD CHARACTER OVER IT
                LA 2,1(2)         BUMP R2
                LA 3,1(3)         BUMP R3
                S 4,=F'1'         DECREMENT THE LENGTH
                IF (CR,3,EQ,1)    IF END OF PAD
```

Figure 17 (Part 20 of 39). Assembler Routine, USERSVC

```
                LA 3,PAD            R3 -> PAD
              ENDIF
            ENDDO
*
*                                   DEBUG MESSAGE
*
            IF (CLI,DBG,EQ,C'Y')
*
              MVC WTO(WTOE-WTOC),WTOC COPY WTO LIST TO DSA
              MVC WTO+4(11),=C'GETPWD PAD='
              L 1,=F'8'             R1 = 8 (LENGTH OF INPUT)
              LA 2,PAD              R2 -> PAD
              LA 3,WTO+15           R3 -> MESSAGE (OUTPUT)
              CALL HEXCONV          CONVERT TO HEX
              WTO MF=(E,WTO)
*
              MVC WTO(WTOE-WTOC),WTOC COPY WTO LIST TO DSA
              MVC WTO+4(15),=C'GETPWD USRDATA='
              L 1,USRDATAL          R1 = USRDATAL
              LA 2,USRDATA          R2 -> USRDATA
              LA 3,WTO+19           R3 -> MESSAGE (OUTPUT)
              CALL HEXCONV          CONVERT TO HEX
              WTO MF=(E,WTO)
*
              MVC WTO(WTOE-WTOC),WTOC COPY WTO LIST TO DSA
              MVC WTO+4(17),=C'GETPWD PARAMETER='
              L 1,=F'8'             R1 = 8 (LENGTH OF INPUT)
              L 2,PTRPLIST            R2 -> PLIST
              L 2,4(2)                R2 -> ENCRYPTED PASSWORD
              LA 3,WTO+21           R3 -> MESSAGE (OUTPUT)
              CALL HEXCONV          CONVERT TO HEX
              WTO MF=(E,WTO)
            ENDIF
          ENDIF
*
*                           *----------------------------------*
*                            GET VERSION OF CURRENT SECRET KEY
*                           *----------------------------------*
*
          MVC CLASS,=CL8'$DCERACF'
*
*                                   BUILD PROFILE NAME
*
          MVC ENTITY,=CL256' '      BLANK ENTITY FIELD
          L 2,16                    R2 -> CVT
          MVC ENTITY(8),CVTSNAME-CVTMAP(2) COPY SYS NAME TO ENTITY
          LA 2,ENTITY               R2 -> ENTITY FIELD
          DO WHILE=(CLI,0(2),NE,C' ') FIND FIRST BLANK
            LA 2,1(2)               BUMP R2
          ENDDO
          MVC 0(8,2),=C'.CURRENT'   MOVE SUFFIX TO ENTITY
*
*                                   ISSUE RACROUTE EXTRACT
*
          MVC RAC04(RAC04E-RAC04C),RAC04C COPY RACROUTE PARAMETERS
          RACROUTE REQUEST=EXTRACT,                                   X
            TYPE=EXTRACT,                                             X
            CLASS=CLASS,                                              X
            ENTITY=ENTITY,                                            X
            FIELDS=INSTDATA,                                          X
            SEGMENT=BASE,                                             X
            WORKA=RACWORKA,                                           X
            RELEASE=1.9.2,                                            X
            MF=(E,RAC04)
          LR 2,1                    SAVE A(RESULT)
          ST 2,SAVEBUF              SAVE A(EXTRACT BUFFER)
*
*                                   TEST RACROUTE EXTRACT RESULT
*
          IF (LTR,15,15,NZ)         IF AN ERROR
```

Figure 17 (Part 21 of 39). Assembler Routine, USERSVC

```
              MVC RC,=F'12'           SET RETURN CODE
              ST 15,DBGR15
              LA 1,RAC04
              MVC DBGMSG,=CL40'GETPWD   - GET CURRENT'
              CALL DBGMSG2
            ELSE
*
*                                     COPY BUFFER
*
              LR 4,2                  R4 -> EXTRACT RESULT
              AH 4,EXTWOFF-EXTWKEA(2) R4 = R4 + OFFSET TO INSTDATA
              L 5,0(4)                R5 = L(INSTDATA)
              LA 5,4(5)               R5 = L(INSTDATA) + LENGTH FIELD
              LA 6,KEYLEN             R6 -> KEYLEN
              LR 7,5                  R7 = LENGTH TO MOVE
              MVCL 6,4                COPY RESULTS
*
*                                     FREE THE EXTRACT BUFFER
*
              L 2,SAVEBUF             R2 -> EXTRACT BUFFER
              XR 3,3                  ZERO OUT R3
              XR 4,4                  ZERO OUT R4
              USING EXTWKEA,2         BASE THE AREA ON R2
              ICM 3,1,EXTWSP          MOVE THE AREA SUBPOOL INTO R3
              ICM 4,7,EXTWLN          MOVE THE AREA LENGTH INTO R4
              DROP 2                  DROP BASING ON REGISTER 2
              FREEMAIN R,LV=(4),A=(2),SP=(3) FREE EXTRACT BUFFER
*
*                                *-----------------------------------*
*                                 RE-ENCRYPT IF USRDATA AND CURRENT
*                                 KEY VERSIONS ARE DIFFERENT
*                                *-----------------------------------*
*
              IF (CLC,KEYVER,NE,KEYVERA) IF VERSIONS DIFFERENT
                CALL SETPWD
              ENDIF
            ENDIF
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ENDIF
 ENDIF
*
*                                *-----------------------------------*
*                                 RETURN
*                                *-----------------------------------*
*
 L     13,4(13)                        R13 -> HSA
 RETURN (14,12)                        RETURN
*
*--------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - EXTRACT DCE PRINCIPAL NAME'
*--------------------------------------------------------------------*
*
GETPRIN DS OH                           GETPRIN SUBROUTINE
 ENTRY GETPRIN
*
*                                *-----------------------------------*
*                                 SAVE REGISTERS AND CHAIN SAVE AREAS
*                                *-----------------------------------*
*
 SAVE  (14,12)                         SAVE REGISTERS
 LA    2,LSA6                          R2 -> LSA 6
 ST    2,8(13)                         LSA IN HSA
 ST    13,4(2)                         HSA IN LSA
 LR    13,2                            LSA IN 13
 LR 11,15                              LOAD BASE
 USING GETPRIN,11                      ESTABLISH ADDRESSING
 MVC RC,=F'0'                          RETURN CODE = 0
```

Figure 17 (Part 22 of 39). Assembler Routine, USERSVC

```
*
*                               *---------------------------------*
*                                FIND CURRENT TCB
*                               *---------------------------------*
*
 L 2,16                              R2 -> CVT
 L 2,CVTTCBP-CVTMAP(2)               R2 -> A(NEXT TCB, CURRENT TCB)
 L 2,4(2)                            R2 -> CURRENT TCB
*
*                               *---------------------------------*
*                                FIND ACEE (TCB OR ASXB)
*                               *---------------------------------*
*
 L 2,TCBSENV-TCB(2)                  R2 -> ACEE
 IF (C,2,EQ,=F'0')                   IS THERE NO ACEE IN TCBSENV
   L 2,16                            R2 -> CVT
   L 2,CVTTCBP-CVTMAP(2)             R2 -> A(NEXT TCB, CURRENT TCB)
   L 2,12(2)                         R2 -> CURRENT ASCB
   L 2,ASCBASXB-ASCB(2)              R2 -> ASXB
   L 2,ASXBSENV-ASXB(2)              R2 -> ACEE
 ENDIF
 ST 2,PTRACEE                        SAVE PTR TO ACEE
*
*                               *---------------------------------*
*                                GET RACF USERID FROM ACEE
*                               *---------------------------------*
*
 MVC USERID,ACEEUSER-ACEE(2)         COPY USERID FROM ACEE
*
*                               *---------------------------------*
*                                GET USRDATA
*                               *---------------------------------*
*
*
 MVC CLASS,=CL8'USER'
 MVC RAC04(RAC04E-RAC04C),RAC04C     COPY RACROUTE PARAMETERS
 RACROUTE REQUEST=EXTRACT,                                          X
               TYPE=EXTRACT,                                        X
               CLASS=CLASS,                                         X
               ENTITY=USERID+1,                                     X
               FIELDS=USRDATAC,                                     X
               SEGMENT=BASE,                                        X
               WORKA=RACWORKA,                                      X
               RELEASE=1.9.2,                                       X
               MF=(E,RAC04)
 LR 2,1                              SAVE A(RESULT)
 ST 2,SAVEBUF                        SAVE A(EXTRACT BUFFER)
*
*                                    DID THE EXTRACT WORK
*
 IF (LTR,15,15,NZ)                   IF AN ERROR
   MVC RC,=F'12'                     SET RETURN CODE
   ST 15,DBGR15
   LA 1,RAC04
   MVC DBGMSG,=CL40'GETPRIN - GET USRDATA'
   CALL DBGMSG2
 ELSE
*
*                                    COPY THE BUFFER
*
   LR 4,2                            R4 -> EXTRACT RESULT
   AH 4,EXTWOFF-EXTWKEA(2)           R4 = R4 + OFFSET TO USRDATA
   LA 4,4(4)                         R4 -> FIRST USRDATA
   L  5,0(4)                         R5 = L(USRDATA)
   LA 5,4(5)                         R5 = L(USRDATA) + LENGTH FIELD
   LA 6,USRDATAL                     R6 -> USRDATAL
   LR 7,5                            R7 = LENGTH TO MOVE
   MVCL 6,4                          COPY RESULTS
*
*                                    FREE THE EXTRACT BUFFER
```

Figure 17 (Part 23 of 39). Assembler Routine, USERSVC

```
*
   L 2,SAVEBUF                        R2 -> EXTRACT BUFFER
   XR 3,3                             ZERO OUT R3
   XR 4,4                             ZERO OUT R4
   USING EXTWKEA,2                    BASE THE AREA ON R2
   ICM 3,1,EXTWSP                     MOVE THE AREA SUBPOOL INTO R3
   ICM 4,7,EXTWLN                     MOVE THE AREA LENGTH INTO R4
   DROP 2                             DROP BASING ON REGISTER 2
   FREEMAIN R,LV=(4),A=(2),SP=(3)     FREE EXTRACT BUFFER
*
*                                *-----------------------------------*
*                                 FIND DCE KEYWORD IN USRDATA
*                                *-----------------------------------*
*
   MVI FOUND,C'N'                     FOUND = NO
   LA 3,USRDATA                       R3 -> USRDATA
   IF (CLC,USRDATAL,NE,=F'0')         IS THERE ANY USRDATA
     IF (CLC,0(4,3),EQ,=C'DCE(')      IF DCE KEYWORD FOUND
       MVI FOUND,C'Y'                 FOUND DCE KEYWORD
     ENDIF
   ENDIF
   IF (CLI,FOUND,EQ,C'N')             IF DCE KEYWORD NOT FOUND
     MVC RC,=F'8'                     RETURN WITH ERROR
   ELSE
*
*                                *-----------------------------------*
*                                 CHECK PRINCIPAL PRINCIPAL PRESENT
*                                *-----------------------------------*
*
     IF (CLI,USRDATA+4,EQ,X'00')      IF PRINCIPAL NOT PRESENT
       MVC RC,=F'8'                   INDICATE WRONG DCE DATA
     ELSE
*
*                                *-----------------------------------*
*                                 COPY PRINCIPAL TO PARAMETER
*                                *-----------------------------------*
*
       XR 1,1                           R1 = 0
       IC 1,USRDATA+4                   R1 = OFFSET TO PRINCIPAL
       LA 2,USRDATA(1)                  R2 -> PRINCIPAL NAME IN USRDATA
       L 3,PTRPLIST                     R3 -> CALLER'S PARAMETER LIST
       L 3,4(3)                         R3 -> CALLER'S PARAMETER
       DO WHILE=(CLI,0(2),NE,X'00')     DO WHILE NOT END OF STRING
         MVC 0(1,3),0(2)                COPY CHARACTER
         LA 2,1(2)                      BUMP R2 (INPUT)
         LA 3,1(3)                      BUMP R3 (OUTPUT)
       ENDDO
       MVI 0(3),X'00'                   STRING TERMINATOR
     ENDIF
   ENDIF
 ENDIF
*
*                                *-----------------------------------*
*                                 RETURN
*                                *-----------------------------------*
*
 L     13,4(13)                         R13 -> HSA
 RETURN (14,12)                         RETURN
*
*--------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - SET DCE PRINCIPAL NAME'
*--------------------------------------------------------------------*
*
SETPRIN  DS 0H                           SETPRIN SUBROUTINE
 ENTRY SETPRIN
*
*                                *-----------------------------------*
*                                 SAVE REGISTERS AND CHAIN SAVE AREAS
*                                *-----------------------------------*
*
```

Figure 17 (Part 24 of 39). Assembler Routine, USERSVC

```
 SAVE (14,12)                          SAVE REGISTERS
 LA 2,LSA7                             R2 -> LSA 7
 ST 2,8(13)                            LSA IN HSA
 ST 13,4(2)                            HSA IN LSA
 LR 13,2                               LSA IN 13
 LR 11,15                              LOAD BASE
 USING SETPRIN,11                      ESTABLISH ADDRESSING
 MVC RC,=F'0'                          RETURN CODE = 0
*
*                                  *-----------------------------------*
*                                   CHECK AUTHORISATION
*                                  *-----------------------------------*
*
 MVI CLASS,X'08'
 MVC CLASS+1(8),=CL8'$DCERACF'         CLASS = $DCERACF
 MVC ENTITY,=CL256' '                  ENTITY = BLANKS
 MVC ENTITY(13),=C'ADMINISTRATOR'      ENTITY = ADMINISTRATOR
 MVC RAC03(RAC03E-RAC03C),RAC03C       COPY RACROUTE PARAMETER LIST
 RACROUTE REQUEST=AUTH,                                              X
               WORKA=RACWORKA,                                       X
               CLASS=CLASS,                                          X
               ENTITY=(ENTITY),                                      X
               ATTR=ALTER,                                           X
               SYSTEM=NO,                                            X
               RELEASE=1.9.2,                                        X
               MF=(E,RAC03)
 IF (LTR,15,15,NZ)                     IF FAILED
   MVC RC,=F'16'                       SET RETURN CODE
   ST 15,DBGR15
   LA 1,RAC03
   MVC DBGMSG,=CL40'SETPRIN  - AUTH'
   CALL DBGMSG2
 ELSE
*
*                                  *-----------------------------------*
*                                   GET RACF USERID FROM PARAMETERS
*                                  *-----------------------------------*
*
   L 2,PTRPLIST                        R2 -> CALLER'S PARAMETER LIST
   L 2,4(2)                            R2 -> CALLER'S PARAMETER
   MVC USERID,=CL9' '                  USERID = BLANKS
   LA 3,USERID                         R3 -> USERID
   DO WHILE=(CLI,0(2),NE,X'00')        DO WHILE NOT NULL
     MVC 0(1,3),0(2)                   COPY CHARACTER
     OI 0(3),X'40'                     FORCE UPPERCASE
     LA 2,1(2)                         BUMP R2 (INPUT)
     LA 3,1(3)                         BUMP R3 (OUTOUT)
   ENDDO
*
*                                  *-----------------------------------*
*                                   BUILD USRDATA
*                                  *-----------------------------------*
*
   MVC USRDATA(4),=C'DCE('             MOVE KEYWORD
   MVC USRDATA+4(3),=X'070000'         INITIALISE OFFSETS
   L 2,PTRPLIST                        R2 -> CALLER'S PARAMETER LIST
   L 2,8(2)                            R2 -> PRINCIPAL NAME
   LA 3,USRDATA+7                      R3 -> OUTPUT
   DO WHILE=(CLI,0(2),NE,X'00')        DO WHILE NOT NULL
     MVC 0(1,3),0(2)                   MOVE CHARACTER
     LA 2,1(2)                         BUMP R2
     LA 3,1(3)                         BUMP R2
   ENDDO
   MVI 0(3),X'00'                      INSERT NULL
   LA 3,1(3)                           BUMP R3
   MVI 0(3),C')'                       INSERT END BRACKET
   LA 3,1(3)                           BUMP R3
*
   LA 4,USRDATA                        R4 -> BEGINNING OF USRDATA
   SR 3,4                              R3 = LENGTH OF USRDATA
```

Figure 17 (Part 25 of 39). Assembler Routine, USERSVC

```
   ST 3,USRDATAL                        STORE LENGTH IN USRDATAL
*
*                                *-----------------------------------*
*                                 UPDATE USRDATA IN RACF PROFILE
*                                *-----------------------------------*
*
   MVC CLASS,=CL8'USER'
   MVC RAC06(RAC06E-RAC06C),RAC06C     COPY MACRO TO WORK AREA
   RACROUTE REQUEST=EXTRACT,                                          X
               TYPE=REPLACE,                                          X
               ENTITY=USERID,                                         X
               CLASS=CLASS,                                           X
               FIELDS=USRDATAC,                                       X
               SEGDATA=USRDATAL,                                      X
               SEGMENT=BASE,                                          X
               WORKA=RACWORKA,                                        X
               RELEASE=1.9.2,                                         X
               MF=(E,RAC06)
   IF (LTR,15,15,NZ)                   IF AN ERROR
     MVC RC,=F'28'                     SET RETURN CODE
     ST 15,DBGR15
     LA 1,RAC06
     MVC DBGMSG,=CL40'SETPRIN  - UPDATE USRDATA'
     CALL DBGMSG2
   ENDIF
 ENDIF
*
*                                *-----------------------------------*
*                                 RETURN
*                                *-----------------------------------*
*
 L     13,4(13)                       R13 -> HSA
 RETURN (14,12)                       RETURN
*
*---------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - LIST USRDATA'
*---------------------------------------------------------------------*
*
GETUSR   DS OH                         GETUSR SUBROUTINE
 ENTRY GETUSR
*
*                                *-----------------------------------*
*                                 SAVE REGISTERS AND CHAIN SAVE AREAS
*                                *-----------------------------------*
*
 SAVE (14,12)                         SAVE REGISTERS
 LA 2,LSA8                            R2 -> LSA 8
 ST 2,8(13)                           LSA IN HSA
 ST 13,4(2)                           HSA IN LSA
 LR 13,2                              LSA IN 13
 LR 11,15                             LOAD BASE
 USING GETUSR,11                      ESTABLISH ADDRESSING
*
*                                *-----------------------------------*
*                                 CHECK AUTHORISATION
*                                *-----------------------------------*
*
 MVI CLASS,X'08'
 MVC CLASS+1(8),=CL8'$DCERACF'        CLASS = $DCERACF
 MVC ENTITY,=CL256' '                 ENTITY = BLANKS
 MVC ENTITY(13),=C'ADMINISTRATOR'     ENTITY = ADMINISTRATOR
 MVC RAC03(RAC03E-RAC03C),RAC03C      COPY RACROUTE PARAMETER LIST
 RACROUTE REQUEST=AUTH,                                             X
               WORKA=RACWORKA,                                      X
               CLASS=CLASS,                                         X
               ENTITY=(ENTITY),                                     X
               ATTR=ALTER,                                          X
               SYSTEM=NO,                                           X
               RELEASE=1.9.2,                                       X
               MF=(E,RAC03)
```

Figure 17 (Part 26 of 39). Assembler Routine, USERSVC

```
 IF (LTR,15,15,NZ)                    IF FAILED
   MVC RC,=F'16'                      SET RETURN CODE
   ST 15,DBGR15
   LA 1,RAC03
   MVC DBGMSG,=CL40'GETUSR  - AUTH'
   CALL DBGMSG2
 ELSE
*
*                               *----------------------------------*
*                                GET RACF USERID
*                               *----------------------------------*
*
   L 2,PTRPLIST                       R2 -> CALLER'S PARAMETER LIST
   L 2,4(2)                           R2 -> CALLER'S PARAMETER
   MVC USERID,=CL9' '                 USERID = BLANKS
   LA 3,USERID                        R3 -> USERID
   DO WHILE=(CLI,0(2),NE,X'00')       DO WHILE NOT NULL
     MVC 0(1,3),0(2)                  COPY CHARACTER
     OI 0(3),X'40'                    FORCE UPPERCASE
     LA 2,1(2)                        BUMP R2 (INPUT)
     LA 3,1(3)                        BUMP R3 (OUTOUT)
   ENDDO
*
*                               *----------------------------------*
*                                GET USRDATA
*                               *----------------------------------*
*
*
   MVC CLASS,=CL8'USER'               CLASS = USER
   MVC RAC04(RAC04E-RAC04C),RAC04C    COPY RACROUTE PARAMETERS
   RACROUTE REQUEST=EXTRACT,                                       X
               TYPE=EXTRACT,                                       X
               CLASS=CLASS,                                        X
               ENTITY=USERID,                                      X
               FIELDS=USRDATAC,                                    X
               SEGMENT=BASE,                                       X
               WORKA=RACWORKA,                                     X
               RELEASE=1.9.2,                                      X
               MF=(E,RAC04)
   LR 2,1                             SAVE A(RESULT)
   ST 2,SAVEBUF                       SAVE A(EXTRACT BUFFER)
*
*                                     TEST RESULT
*
   IF (LTR,15,15,NZ)                  IF AN ERROR
     MVC RC,=F'12'                    SET RETURN CODE
     ST 15,DBGR15
     LA 1,RAC04
     MVC DBGMSG,=CL40'GETUSR  - GET USRDATA'
     CALL DBGMSG2
   ELSE
*
*                                     COPY BUFFER
*
     L 2,SAVEBUF                      R2 -> EXTRACT BUFFER
     LR 4,2                           R4 -> EXTRACT RESULT
     AH 4,EXTWOFF-EXTWKEA(2)          R4 = R4 + OFFSET TO USRDATA
     LA 4,4(4)                        R4 -> FIRST USRDATA
     L 5,0(4)                         R5 = L(USRDATA)
     LA 5,4(5)                        R5 = L(USRDATA) + LENGTH FIELD
     LA 6,USRDATAL                    R6 -> USRDATAL
     LR 7,5                           R7 = LENGTH TO MOVE
     MVCL 6,4                         COPY RESULTS
*
*                                     FREE BUFFER
*
     L 2,SAVEBUF                      R2 -> EXTRACT BUFFER
     XR 3,3                           ZERO OUT R3
     XR 4,4                           ZERO OUT R4
     USING EXTWKEA,2                  BASE THE AREA ON R2
```

Figure 17 (Part 27 of 39). Assembler Routine, USERSVC

```
     ICM 3,1,EXTWSP                   MOVE THE AREA SUBPOOL INTO R3
     ICM 4,7,EXTWLN                   MOVE THE AREA LENGTH INTO R4
     DROP 2                           DROP BASING ON REGISTER 2
     FREEMAIN R,LV=(4),A=(2),SP=(3)   FREE EXTRACT BUFFER
*
*                              *----------------------------------*
*                               COPY TO PARAMETER (CHARACTER)
*                              *----------------------------------*
*
     L 3,PTRPLIST                     R3 -> CALLER'S PARAMETER LIST
     L 3,8(3)                         R3 -> FORMAT REQUESTED
     IF (CLI,0(3),EQ,C'C'),OR,(CLI,0(3),EQ,C'c') IF CHARACTER
       LA 2,USRDATA                   R2 -> USRDATA
       L 3,PTRPLIST                   R3 -> CALLER'S PARAMETER LIST
       L 3,12(3)                      R3 -> CALLER'S WORKAREA
       L 1,USRDATAL                   R3 = USRDATA LENGTH
       DO WHILE=(LTR,1,1,NZ)          DO WHILE THERE IS DATA
         MVC 0(1,3),0(2)              COPY CHARACTER
         IF (CLI,0(3),EQ,X'00')       IF NULL CHARACTER
           MVI 0(3),C' '              BLANK IT OUT
         ENDIF
         S 1,=F'1'                    DECREMENT R1
         LA 2,1(2)                    BUMP R2
         LA 3,1(3)                    BUMP R3
       ENDDO
       MVI 0(3),X'00'                 INSERT NULL TERMINATOR
     ELSE
*
*                              *----------------------------------*
*                               COPY TO PARAMETER (HEX)
*                              *----------------------------------*
*
       L  1,USRDATAL                  R1 = USRDATAL
       LA 2,USRDATA                   R2 -> USRDATA
       L 3,PTRPLIST                   R3 -> CALLER'S PARAMETER LIST
       L 3,12(3)                      R3 -> CALLER'S WORKAREA
       CALL HEXCONV
       SLL 1,1                        R1 -> DOUBLE THE LENGTH
       LA 3,2(1,3)                    R3 -> END OF HEX DATA
       MVI 0(3),X'00'                 INSERT NULL DELIMITER
     ENDIF
   ENDIF
 ENDIF
*
*                              *----------------------------------*
*                               RETURN
*                              *----------------------------------*
*
 L 13,4(13)                           R13 -> HSA
 RETURN (14,12)                       RETURN
*
*--------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - EXTRACT RACF SYSID'
*--------------------------------------------------------------------*
*
GETSYSID DS OH                         GETSYSID SUBROUTINE
 ENTRY GETSYSID
*
*                              *----------------------------------*
*                               SAVE REGISTERS AND CHAIN SAVE AREAS
*                              *----------------------------------*
*
 SAVE (14,12)                         SAVE REGISTERS
 LA 2,LSA9                            R2 -> LSA 9
 ST 2,8(13)                           LSA IN HSA
 ST 13,4(2)                           HSA IN LSA
 LR 13,2                              LSA IN 13
 LR 11,15                             LOAD BASE
 USING GETSYSID,11                    ESTABLISH ADDRESSING
 MVC RC,=F'0'                         RETURN CODE = 0
```

Figure 17 (Part 28 of 39). Assembler Routine, USERSVC

```
*
*                               *-----------------------------------*
*                                COPY TO CALLER'S PARAMETER
*                               *-----------------------------------*
*
 L 2,16                              R2 -> CVT
 MVC SYSID(9),=XL9'00'               CLEAR SYSID
 MVC SYSID(8),CVTSNAME-CVTMAP(2)     COPY SYSTEM NAME TO SYSID
 LA 2,SYSID                          R2 -> SYSID
 L 3,PTRPLIST                        R3 -> CALLER'S PARAMETER LIST
 L 3,4(3)                            R3 -> CALLER'S PARAMETER
 DO WHILE=(CLI,0(2),NE,X'00')        DO WHILE NOT END OF STRING
   MVC 0(1,3),0(2)                   COPY CHARACTER
   IF (CLI,0(3),EQ,C' ')             IF A BLANK
     MVI 0(3),X'00'                  TERMINATE STRING
   ENDIF
   LA 2,1(2)                         BUMP R2 (INPUT)
   LA 3,1(3)                         BUMP R3 (OUTPUT)
 ENDDO
 MVI 0(3),X'00'                      STRING TERMINATOR
*
*                               *-----------------------------------*
*                                RETURN
*                               *-----------------------------------*
*
 L     13,4(13)                      R13 -> HSA
 RETURN (14,12)                      RETURN
*
*--------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - SET DCE UUID'
*--------------------------------------------------------------------*
*
SETUUID  DS 0H                        SETUUID SUBROUTINE
 ENTRY SETUUID
*
*                               *-----------------------------------*
*                                SAVE REGISTERS AND CHAIN SAVE AREAS
*                               *-----------------------------------*
*
 SAVE (14,12)                        SAVE REGISTERS
 LA 2,LSA10                          R2 -> LSA 10
 ST 2,8(13)                          LSA IN HSA
 ST 13,4(2)                          HSA IN LSA
 LR 13,2                             LSA IN 13
 LR 11,15                            LOAD BASE
 USING SETUUID,11                    ESTABLISH ADDRESSING
 MVC RC,=F'0'                        RETURN CODE = 0
*
*                               *-----------------------------------*
*                                CHECK AUTHORISATION
*                               *-----------------------------------*
*
 MVI CLASS,X'08'
 MVC CLASS+1(8),=CL8'$DCERACF'       CLASS = $DCERACF
 MVC ENTITY,=CL256' '                ENTITY = BLANKS
 MVC ENTITY(13),=C'ADMINISTRATOR'    ENTITY = ADMINISTRATOR
 MVC RAC03(RAC03E-RAC03C),RAC03C     COPY RACROUTE PARAMETER LIST
 RACROUTE REQUEST=AUTH,                                          X
               WORKA=RACWORKA,                                   X
               CLASS=CLASS,                                      X
               ENTITY=(ENTITY),                                  X
               ATTR=ALTER,                                       X
               SYSTEM=NO,                                        X
               RELEASE=1.9.2,                                    X
               MF=(E,RAC03)
 IF (LTR,15,15,NZ)                   IF FAILED
   MVC RC,=F'16'                     SET RETURN CODE
   ST 15,DBGR15
   LA 1,RAC03
   MVC DBGMSG,=CL40'SETUUID - AUTH'
```

Figure 17 (Part 29 of 39). Assembler Routine, USERSVC

```
   CALL DBGMSG2
 ELSE
*
*                               *-----------------------------------*
*                                GET UUID FROM PARAMETERS
*                               *-----------------------------------*
*
*
   L 2,PTRPLIST                      R2 -> PARAMETER LIST
   L 3,4(2)                          R3 -> UUID STRING
   LA 4,ENTITY                       R4 -> ENTITY
   MVC ENTITY,=CL256' '              COPY BLANKS
   DO WHILE=(CLI,0(3),NE,X'00')      DO WHILE NOT NULL TERMINATOR
     MVC 0(1,4),0(3)                 COPY CHARACTER
     LA 3,1(3)                       BUMP R3
     LA 4,1(4)                       BUMP R4
   ENDDO
*                               *-----------------------------------*
*                                  GET RACF USERID FROM PARAMETERS
*                               *-----------------------------------*
*
   XR 1,1                            R1 = 0 (COUNTER)
   L 2,PTRPLIST                      R2 -> PARAMETER LIST
   L 3,8(2)                          R3 -> CALLER'S USERID PARM
   LA 4,USERID+1                     R4 -> ENTITY
   XC USERID,USERID                  MAKE ALL NULLS
   DO WHILE=(CLI,0(3),NE,X'00')      DO WHILE NOT NULL TERMINATOR
     MVC 0(1,4),0(3)                 COPY CHARACTER
     LA 1,1(1)                       BUMP R1 (COUNT)
     LA 3,1(3)                       BUMP R3 (INPUT)
     LA 4,1(4)                       BUMP R4 (OUTPUT)
   ENDDO
   STC 1,USERID                      STORE LENGTH
*
*                               *-----------------------------------*
*                                DOES UUID PROFILE EXIST
*                               *-----------------------------------*
*
   MVC SVUSERID,USERID               SAVE USERID
   MVC SVENTITY,ENTITY               SAVE ENTITY
   MVC SAVPLIST,PTRPLIST             SAVE CALLER'S PLIST POINTER
*
   L 2,PTRPLIST                      R2 -> PARAMETER LIST
   L 1,4(2)                          R3 -> UUID STRING
   ST 1,PLIST+4                      STORE IN PARAMETER LIST
   LA 1,DCERACU                      R1 -> DCERACU
   ST 1,PLIST+8                      STORE IN PARAMETER LIST
   OI PLIST+8,X'80'                  SET END OF LIST
   LA 1,PLIST                        R1 -> PARAMETER LIST
   ST 1,PTRPLIST                     STORE OVER CALLER'S PLIST PTR
   CALL GETUUID                      GET USERID FROM UUID PROFILE
*
   MVC PTRPLIST,SAVPLIST             RESTORE CALLER'S PLIST POINTER
   MVC ENTITY,SVENTITY               RESTORE ENTITY
   MVC USERID,SVUSERID               RESTORE USERID
*
*                               *-----------------------------------*
*                                  DELETE ANY EXISTING PROFILE
*                               *-----------------------------------*
*
   IF (CLC,RC,EQ,=F'0')              IF RC FROM GETUUID = 0
     MVI CLASS,X'08'
     MVC CLASS+1(8),=CL8'$DCERACF'
     MVC RAC07(RAC07E-RAC07C),RAC07C   COPY RACROUTE PARAMETERS
     RACROUTE REQUEST=DEFINE,                                      X
               TYPE=DELETE,                                        X
               WORKA=RACWORKA,                                     X
               CLASS=CLASS,                                        X
               ENTITY=ENTITY,                                      X
               RELEASE=1.9.2,                                      X
```

Figure 17 (Part 30 of 39). Assembler Routine, USERSVC

```
              MF=(E,RAC07)
   ENDIF
   MVC RC,=F'0'                        RESET RC
*
*                                  *-----------------------------------*
*                                   DEFINE PROFILE
*                                  *-----------------------------------*
*
   MVI CLASS,X'08'
   MVC CLASS+1(8),=CL8'$DCERACF'
   MVC RAC08(RAC08E-RAC08C),RAC08C     COPY RACROUTE PARAMETERS
   RACROUTE REQUEST=DEFINE,                                           X
              TYPE=DEFINE,                                            X
              WORKA=RACWORKA,                                         X
              CLASS=CLASS,                                            X
              ENTITY=ENTITY,                                          X
              DATA=USERID,                                            X
              RELEASE=1.9.2,                                          X
              MF=(E,RAC08)
   IF (LTR,15,15,NZ)                   IF FAILED
     MVC RC,=F'8'                      SET RETURN CODE
     ST 15,DBGR15
     LA 1,RAC08
     MVC DBGMSG,=CL40'SETUUID - DEFINE UUID PROFILE'
     CALL DBGMSG2
   ENDIF
 ENDIF
*                                  *-----------------------------------*
*                                   RETURN
*                                  *-----------------------------------*
*
 L     13,4(13)                      R13 -> HSA
 RETURN (14,12)                      RETURN
*
*--------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - GET DCE UUID'
*--------------------------------------------------------------------*
*
GETUUID  DS 0H                        GETUUID SUBROUTINE
 ENTRY GETUUID
*
*                                  *-----------------------------------*
*                                   SAVE REGISTERS AND CHAIN SAVE AREAS
*                                  *-----------------------------------*
*
 SAVE (14,12)                        SAVE REGISTERS
 LA 2,LSA11                          R2 -> LSA 11
 ST 2,8(13)                          LSA IN HSA
 ST 13,4(2)                          HSA IN LSA
 LR 13,2                             LSA IN 13
 LR 11,15                            LOAD BASE
 USING GETUUID,11                    ESTABLISH ADDRESSING
 MVC RC,=F'0'                        RETURN CODE = 0
*
*                                  *-----------------------------------*
*                                   GET UUID FROM PARAMETERS
*                                  *-----------------------------------*
*
*
 L 2,PTRPLIST                        R2 -> PARAMETER LIST
 L 3,4(2)                            R3 -> UUID STRING
 LA 4,ENTITY                         R4 -> ENTITY
 MVC ENTITY,=CL256' '                COPY BLANKS
 DO WHILE=(CLI,0(3),NE,X'00')        DO WHILE NOT NULL TERMINATOR
   MVC 0(1,4),0(3)                   COPY CHARACTER
   LA 3,1(3)                         BUMP R3
   LA 4,1(4)                         BUMP R4
 ENDDO
*
*                                  *-----------------------------------*
```

Figure 17 (Part 31 of 39). Assembler Routine, USERSVC

```
*                                EXTRACT INSTDATA FIELD
*                               *----------------------------------*
*
 MVC CLASS(8),=CL8'$DCERACF'
 MVC RAC04(RAC04E-RAC04C),RAC04C       COPY RACROUTE PARAMETERS
 RACROUTE REQUEST=EXTRACT,                                          X
               TYPE=EXTRACT,                                        X
               CLASS=CLASS,                                         X
               ENTITY=ENTITY,                                       X
               FIELDS=INSTDATA,                                     X
               SEGMENT=BASE,                                        X
               WORKA=RACWORKA,                                      X
               RELEASE=1.9.2,                                       X
               MF=(E,RAC04)
 LR 2,1                                SAVE A(RESULT)
 ST 2,SAVEBUF                          SAVE A(EXTRACT BUFFER)
*
*                                      TEST RACROUTE EXTRACT RESULT
*
 IF (LTR,15,15,NZ)                     IF AN ERROR
   MVC RC,=F'8'                        SET RETURN CODE
   ST 15,DBGR15
   LA 1,RAC04
   MVC DBGMSG,=CL40'GETUUID  - GET DATA   '
   CALL DBGMSG2
 ELSE
*
*                                      COPY BUFFER
*
   LR 4,2                              R4 -> EXTRACT RESULT
   AH 4,EXTWOFF-EXTWKEA(2) R4 = R4 + OFFSET TO INSTDATA
   L 5,0(4)                            R5 = L(INSTDATA)
   LA 5,4(5)                           R5 = L(INSTDATA) + LENGTH FIELD
   LA 6,DATALEN                        R6 -> DATALEN
   LR 7,5                              R7 = LENGTH TO MOVE
   MVCL 6,4                            COPY RESULTS
*
*                                      FREE THE EXTRACT BUFFER
*
   L 2,SAVEBUF                         R2 -> EXTRACT BUFFER
   XR 3,3                              ZERO OUT R3
   XR 4,4                              ZERO OUT R4
   USING EXTWKEA,2                     BASE THE AREA ON R2
   ICM 3,1,EXTWSP                      MOVE THE AREA SUBPOOL INTO R3
   ICM 4,7,EXTWLN                      MOVE THE AREA LENGTH INTO R4
   DROP 2                              DROP BASING ON REGISTER 2
   FREEMAIN R,LV=(4),A=(2),SP=(3)      FREE EXTRACT BUFFER
*
*                               *----------------------------------*
*                                COPY USERID TO CALLER'S PARM
*                               *----------------------------------*
*
   L 1,DATALEN                         R1 = LENGTH OF THE DATA
   LA 2,DATA                           R2 -> USERID FROM PROFILE
   L 3,PTRPLIST                        R3 -> CALLER'S PARAMETER LIST
   L 3,8(3)                            R3 -> CALLER'S PARAMETER
   DO WHILE=(LTR,1,1,NZ)               DO WHILE THERE IS DATA
     MVC 0(1,3),0(2)                   COPY CHARACTER
     S 1,=F'1'                         DECREMENT COUNT
     LA 2,1(2)                         BUMP R2 (INPUT)
     LA 3,1(3)                         BUMP R3 (OUTPUT)
   ENDDO
   MVI 0(3),X'00'                      INSERT NULL TERMINATOR
 ENDIF
*
*                               *----------------------------------*
*                                RETURN
*                               *----------------------------------*
*
 L     13,4(13)                        R13 -> HSA
```

Figure 17 (Part 32 of 39). Assembler Routine, USERSVC

```
 RETURN (14,12)                        RETURN
*
*-------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - DBGMSG1 SUBROUTINE'
*-------------------------------------------------------------------*
*
DBGMSG1 DS OH                          DBGMSG1 SUBROUTINE
 ENTRY DBGMSG1
*
*                                  *----------------------------------*
*                                   SAVE REGISTERS AND CHAIN SAVE AREAS
*                                  *----------------------------------*
*
 SAVE (14,12)                          SAVE REGISTERS
 LA 14,LSA12                           R2 -> LSA 12
 ST 14,8(13)                           LSA IN HSA
 ST 13,4(14)                           HSA IN LSA
 LR 13,14                              LSA IN 13
 LR 11,15                              LOAD BASE
 USING DBGMSG1,11                      ESTABLISH ADDRESSING
*
 IF (CLI,DBG,EQ,C'Y')
   MVC WTO(WTOE-WTOC),WTOC             COPY WTO LIST TO DSA
   MVC WTO+4(26),=C'RETURNED FROM GETPAC,  RC='
*
   L 1,=F'4'                           R1 = 4 (LENGTH OF INPUT)
   LA 2,RC                             R2 -> RC
   LA 3,WTO+30                         R3 -> MESSAGE (OUTPUT)
   CALL HEXCONV                        CONVERT TO HEX
*
   LA 3,WTO+39
   MVC 0(6,3),=C' UUID='
   LA 2,UUIDSTR                        R2 -> DCE PRINCIPAL NAME
   LA 3,6(3)
   DO WHILE=(CLI,0(2),NE,X'00')        WHILE NOT NULL
     MVC 0(1,3),0(2)                   COPY CHARACTER
     LA 2,1(2)                         BUMP IN
     LA 3,1(3)                         BUMP OUT
   ENDDO
   WTO MF=(E,WTO)
 ENDIF
*
*                                  *----------------------------------*
*                                   RETURN
*                                  *----------------------------------*
*
 L     13,4(13)                        R13 -> HSA
 RETURN (14,12)                        RETURN
*
*-------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - DBGMSG2 SUBROUTINE'
*-------------------------------------------------------------------*
*
DBGMSG2 DS OH                          DBGMSG2 SUBROUTINE
 ENTRY DBGMSG2
*
*                                  *----------------------------------*
*                                   SAVE REGISTERS AND CHAIN SAVE AREAS
*                                  *----------------------------------*
*
 SAVE (14,12)                          SAVE REGISTERS
 LA 14,LSA13                           R2 -> LSA 13
 ST 14,8(13)                           LSA IN HSA
 ST 13,4(14)                           HSA IN LSA
 LR 13,14                              LSA IN 13
 LR 11,15                              LOAD BASE
 USING DBGMSG2,11                      ESTABLISH ADDRESSING
*
 IF (CLI,DBG,EQ,C'Y')
   LR 7,1                              R7 = R1
```

Figure 17 (Part 33 of 39). Assembler Routine, USERSVC

```
   MVC WTO(WTOE-WTOC),WTOC             COPY WTO LIST TO DSA
*
   LA 3,WTO+4                          R3 -> MESSAGE (OUTPUT)
   MVC 0(30,3),DBGMSG
*
   MVC WTO+34(4),=C'R15='
*
   L 1,=F'4'                           R1 = 4 (LENGTH OF INPUT)
   LA 2,DBGR15                         R2 -> SAF RC
   LA 3,WTO+38                         R3 -> MESSAGE (OUTPUT)
   CALL HEXCONV                        CONVERT TO HEX
*
   MVC WTO+46(4),=C' RC='
*
   L 1,=F'4'                           R1 = 4 (LENGTH OF INPUT)
   LR 2,7                              R2 -> RACF RETURN CODE
   LA 3,WTO+50                         R3 -> MESSAGE (OUTPUT)
   CALL HEXCONV                        CONVERT TO HEX
*
   MVC WTO+58(4),=C' RS='
*
   L 1,=F'4'                           R1 = 4 (LENGTH OF INPUT)
   LA 2,4(7)                           R2 -> RACF REASON CODE
   LA 3,WTO+62                         R3 -> MESSAGE (OUTPUT)
   CALL HEXCONV                        CONVERT TO HEX
*
   WTO MF=(E,WTO)
 ENDIF
*
*                                 *-----------------------------------*
*                                  RETURN
*                                 *-----------------------------------*
*
 L     13,4(13)                         R13 -> HSA
 RETURN (14,12)                         RETURN
*
*--------------------------------------------------------------------*
 TITLE 'DCE/RACF USER SVC - HEXCONV SUBROUTINE'
*--------------------------------------------------------------------*
*
* HEXCONV SUBROUTINE
*
* INPUT
*
*      R1 =  COUNT OF CHARACTERS TO BE CONVERTED
*      R2 -> INPUT CHARACTERS
*      R3 -> OUTPUT CHARACTERS
*
*--------------------------------------------------------------------*
HEXCONV DS 0H                           HEXCONV SUBROUTINE
 ENTRY HEXCONV
*
*                                 *-----------------------------------*
*                                  SAVE REGISTERS AND CHAIN SAVE AREAS
*                                 *-----------------------------------*
*
 SAVE (14,12)                           SAVE REGISTERS
 LA 14,LSA14                            R2 -> LSA 14
 ST 14,8(13)                            LSA IN HSA
 ST 13,4(14)                            HSA IN LSA
 LR 13,14                               LSA IN 13
 LR 11,15                               LOAD BASE
 USING HEXCONV,11                       ESTABLISH ADDRESSING
*
*                                 *-----------------------------------*
*                                  CONVERT
*                                 *-----------------------------------*
*
 DO    WHILE=(LTR,1,1,NZ)               DO COUNT TIMES
   XR    4,4                            CLEAR R4
```

Figure 17 (Part 34 of 39). Assembler Routine, USERSVC

```
   IC    4,0(2)                        R4 = WHOLE BYTE
   SRDL  4,4                           R4 = HI 4 BITS, LOW INTO R5
   IC    4,HEX(4)                      R4 = HEX HI 4 BITS
   STC   4,0(3)                        STORE OUTPUT
   XR    4,4                           CLEAR R4
   SLDL  4,4                           R4 = LOW 4 BITS FROM R5
   IC    4,HEX(4)                      R4 = HEX LOW 4 BITS
   STC   4,1(3)                        STORE OUTPUT
   S     1,=F'1'                       DECREMENT R1
   LA    2,1(2)                        BUMP R2
   LA    3,2(3)                        BUMP R3
 ENDDO
*
*                                *-----------------------------------*
*                                 RETURN
*                                *-----------------------------------*
*
 L     13,4(13)                       R13 -> HSA
 RETURN (14,12)                       RETURN
*
*--------------------------------------------------------------------*
 TITLE 'CONSTANTS'
*--------------------------------------------------------------------*
*
CONSTANT DS    OD
HEX      DC    C'0123456789ABCDEF'
INSTDATA DC    A(1)                  NUMBER OF FIELDS
         DC    CL8'INSTDATA'         INSTDATA FIELD
USRDATAC DC    A(1)                  NUMBER OF FIELDS
         DC    CL8'USRDATA'          USRDATA FIELD
BASE     DC    CL8'BASE'             SEGMENT BASE
APPLDCE  DC    CL8'DCERACF'
WTOC     WTO   '                                                    X
                                                                   X
                          ',ROUTCDE=(8,11),DESC=(7),MF=L
WTOE     EQU   *
*
RACO1C   RACROUTE REQUEST=VERIFY,                                  X
               ENVIR=CREATE,                                       X
               WORKA=*-*,                                          X
               USERID=*-*,                                         X
               APPL=*-*,                                           X
               PASSCHK=NO,                                         X
               SYSTEM=NO,                                          X
               RELEASE=1.9.2,                                      X
               MF=L
RACO1E   EQU   *
*
RACO2C   RACROUTE REQUEST=VERIFY,                                  X
               ENVIR=DELETE,                                       X
               WORKA=*-*,                                          X
               SYSTEM=NO,                                          X
               RELEASE=1.9.2,                                      X
               MF=L
RACO2E   EQU   *
*
RACO3C   RACROUTE REQUEST=AUTH,                                    X
               WORKA=*-*,                                          X
               CLASS=*-*,                                          X
               ENTITY=(*-*),                                       X
               RELEASE=1.9.2,                                      X
               MF=L
RACO3E   EQU   *
*
RACO4C   RACROUTE REQUEST=EXTRACT,                                 X
               TYPE=EXTRACT,                                       X
               CLASS=*-*,                                          X
               ENTITY=*-*,                                         X
               FIELDS=*-*,                                         X
               SEGMENT=*-*,                                        X
```

Figure 17 (Part 35 of 39). Assembler Routine, USERSVC

```
              WORKA=*-*,                                              X
              RELEASE=1.9.2,                                          X
              MF=L
RAC04E   EQU   *
*
RAC05C RACROUTE REQUEST=EXTRACT,                                      X
              TYPE=ENCRYPT,                                           X
              ENTITY=*-*,                                             X
              ENCRYPT=(*-*,STDDES),                                   X
              WORKA=*-*,                                              X
              RELEASE=1.9.2,                                          X
              MF=L
RAC05E   EQU   *
*
RAC06C RACROUTE REQUEST=EXTRACT,                                      X
              TYPE=REPLACE,                                           X
              ENTITY=*-*,                                             X
              CLASS=*-*,                                              X
              FIELDS=*-*,                                             X
              SEGDATA=*-*,                                            X
              SEGMENT=*-*,                                            X
              WORKA=*-*,                                              X
              RELEASE=1.9.2,                                          X
              MF=L
RAC06E   EQU   *
*
RAC07C RACROUTE REQUEST=DEFINE,                                       X
              TYPE=DELETE,                                            X
              WORKA=*-*,                                              X
              CLASS=*-*,                                              X
              ENTITY=*-*,                                             X
              RELEASE=1.9.2,                                          X
              MF=L
RAC07E   EQU   *
*
RAC08C RACROUTE REQUEST=DEFINE,                                       X
              TYPE=DEFINE,                                            X
              WORKA=*-*,                                              X
              CLASS=*-*,                                              X
              ENTITY=*-*,                                             X
              DATA=*-*,                                               X
              RELEASE=1.9.2,                                          X
              MF=L
RAC08E   EQU   *
*
RAC09C   RACROUTE REQUEST=VERIFY,                                     X
              ENVIR=CREATE,                                           X
              WORKA=*-*,                                              X
              USERID=*-*,                                             X
              APPL=*-*,                                               X
              PASSCHK=YES,                                            X
              PASSWRD=*-*,                                            X
              SYSTEM=NO,                                              X
              RELEASE=1.9.2,                                          X
              MF=L
RAC09E   EQU   *
*
*--------------------------------------------------------------------*
 TITLE 'LITERALS'
*--------------------------------------------------------------------*
*
 LTORG
*
*--------------------------------------------------------------------*
 TITLE 'DYNAMIC STORAGE AREA'
*--------------------------------------------------------------------*
*
DSA      DSECT
*
LSA      DS    18F                    REGISTER SAVE AREA
```

Figure 17 (Part 36 of 39). Assembler Routine, USERSVC

```
LSA1     DS    18F                    REGISTER SAVE AREA
LSA2     DS    18F                    REGISTER SAVE AREA
LSA3     DS    18F                    REGISTER SAVE AREA
LSA4     DS    18F                    REGISTER SAVE AREA
LSA5     DS    18F                    REGISTER SAVE AREA
LSA6     DS    18F                    REGISTER SAVE AREA
LSA7     DS    18F                    REGISTER SAVE AREA
LSA8     DS    18F                    REGISTER SAVE AREA
LSA9     DS    18F                    REGISTER SAVE AREA
LSA10    DS    18F                    REGISTER SAVE AREA
LSA11    DS    18F                    REGISTER SAVE AREA
LSA12    DS    18F                    REGISTER SAVE AREA
LSA13    DS    18F                    REGISTER SAVE AREA
LSA14    DS    18F                    REGISTER SAVE AREA
SAVEDHSA DS    18F                    SAVED HSA
*
PTRPLIST DS    F                      SAVED PARAMETER LIST ADDRESS
PTRCENV  DS    F                      SAVED A('C' ENVIRONMENT)
PTRTCB   DS    F                      SAVED A(TCB)
PTRACEE  DS    F                      SAVED A(ACEE)
SAVPLIST DS    F                      SAVED PARAMETER LIST ADDRESS
*
CENV     DS    0F                     SAVED C ENVIRONMENT REGISTERS
CENVR12  DS    F                      SAVED C ENVIRONMENT R12
CENVR13  DS    F                      SAVED C ENVIRONMENT R13
*
RC       DS    F                      RETURN CODE
*
PLIST    DS    6F                     PARAMETER LIST
UUIDSTR  DS    CL36,CL1               UUID STRING
USRPRIN  DS    CL128,CL1              ACEE PRINCIPAL NAME
DCEPRIN  DS    CL128,CL1              DCE PRINCIPAL NAME
DCERACU  DS    CL8,CL1                RACF USERID FROM PAC
FOUND    DS    CL1                    FLAG
*
CLASS    DS    CL9                    CLASS FOR REQUEST
*
ENTITY   DS    CL256                  ENTITY FOR REQUEST
SVENTITY DS    CL256                  SAVED ENTITY FOR REQUEST
USERID   DS    CL9                    RACF USERID
SVUSERID DS    CL9                    SAVED RACF USERID
PASSTICK DS    CL9                    RACF PASSTICKET/PASSWORD
APPLPTKT DS    CL8                    RACF PASSTICKET APPLNAME
SAVEBUF  DS    F                      SAVED A(EXTRACT BUFFER)
*
KEYLEN   DS    A(*-*)                 LENGTH OF RESULT
KEY      DS    CL8                    SECRET KEY
KEYVER   DS    CL3                    SECRET KEY VERSION
KEYREST  DS    CL245                  UNUSED BUFFER
*
KEYVERA  DS    CL3                    SECRET KEY VERSION IN ACEE
PWDLEN   DS    F                      SAVED PASSWORD LENGTH
*
USRDATAL DS    A                      LENGTH OF USERDATA
USRDATA  DS    CL256                  USRDATA
*
DATALEN  DS    A                      LENGTH OF DATA
DATA     DS    CL256                  DATA
*
ACCESS   DS    X                      ACCESS
*
PADAREA  DS    0CL9
PADLEN   DS    AL1
PAD      DS    CL8
*
SYSID    DS    0CL9
SYSIDL   DS    AL1
SYSIDD   DS    CL8
*
DBGR15   DS    F
```

Figure 17 (Part 37 of 39). Assembler Routine, USERSVC

```
DBGMSG   DS    CL30
DBG      DS    C
*
WTO      WTO   '                                                      X
                                                                      X
                           ',ROUTCDE=11,DESC=(7),MF=L
*
RAC01    RACROUTE REQUEST=VERIFY,                                     X
               ENVIR=CREATE,                                         X
               WORKA=*-*,                                            X
               USERID=*-*,                                           X
               APPL=*-*,                                             X
               PASSCHK=NO,                                           X
               SYSTEM=NO,                                            X
               RELEASE=1.9.2,                                        X
               MF=L
*
RAC02    RACROUTE REQUEST=VERIFY,                                     X
               ENVIR=DELETE,                                         X
               WORKA=*-*,                                            X
               SYSTEM=NO,                                            X
               RELEASE=1.9.2,                                        X
               MF=L
*
RAC03    RACROUTE REQUEST=AUTH,                                       X
               WORKA=*-*,                                            X
               CLASS=*-*,                                            X
               ENTITY=(*-*),                                         X
               RELEASE=1.9.2,                                        X
               MF=L
*
RAC04    RACROUTE REQUEST=EXTRACT,                                    X
               TYPE=EXTRACT,                                         X
               CLASS=*-*,                                            X
               ENTITY=*-*,                                           X
               FIELDS=*-*,                                           X
               SEGMENT=*-*,                                          X
               WORKA=*-*,                                            X
               RELEASE=1.9.2,                                        X
               MF=L
*
RAC05  RACROUTE REQUEST=EXTRACT,                                     X
               TYPE=ENCRYPT,                                         X
               ENTITY=*-*,                                           X
               ENCRYPT=(*-*,STDDES),                                 X
               WORKA=*-*,                                            X
               RELEASE=1.9.2,                                        X
               MF=L
*
RAC06  RACROUTE REQUEST=EXTRACT,                                     X
               TYPE=REPLACE,                                         X
               ENTITY=*-*,                                           X
               CLASS=*-*,                                            X
               FIELDS=*-*,                                           X
               SEGDATA=*-*,                                          X
               SEGMENT=*-*,                                          X
               WORKA=*-*,                                            X
               RELEASE=1.9.2,                                        X
               MF=L
*
RAC07  RACROUTE REQUEST=DEFINE,                                      X
               TYPE=DELETE,                                          X
               WORKA=*-*,                                            X
               CLASS=*-*,                                            X
               ENTITY=*-*,                                           X
               RELEASE=1.9.2,                                        X
               MF=L
*
RAC08  RACROUTE REQUEST=DEFINE,                                      X
               TYPE=DEFINE,                                          X
```

Figure 17 (Part 38 of 39). Assembler Routine, USERSVC

```
               WORKA=*-*,                                              X
               CLASS=*-*,                                              X
               ENTITY=*-*,                                             X
               DATA=*-*,                                               X
               RELEASE=1.9.2,                                          X
               MF=L
*
*
RACO9    RACROUTE REQUEST=VERIFY,                                      X
               ENVIR=CREATE,                                           X
               WORKA=*-*,                                              X
               USERID=*-*,                                             X
               APPL=*-*,                                               X
               PASSCHK=YES,                                            X
               PASSWRD=*-*,                                            X
               SYSTEM=NO,                                              X
               RELEASE=1.9.2,                                          X
               MF=L
RACWORKA DS    0D,512X
*
DSAE     DS    0D                     END OF DSA
*
*--------------------------------------------------------------------*
 TITLE 'DSECTS'
*--------------------------------------------------------------------*
*
 PRINT NOGEN
 CVT DSECT=YES
 IKJTCB
 IRRPRXTW
 IHAASCB
 IHAASXB
 IHAACEE
 IEESMCA
 END
```

Figure 17 (Part 39 of 39). Assembler Routine, USERSVC

B.3 C Routines

The data set hlq.DCERACF.C contains the following members:

DCERACFU

GETPAC

PTKTLIB

PTKTS

PTSTC

PTSTS

SAVEPW

SAVEPWS

B.3.1 DCERACFU

```
/*********************************************************************/
/*                                                                   */
/* MODULE NAME: DCERACFU                                             */
/*                                                                   */
/* DESCRIPTIVE NAME: DCE/RACF utility                                */
/*                                                                   */
/*  FUNCTION:                                                        */
/*                                                                   */
/*      DCERACFU utility has several functions:                      */
/*                                                                   */
/*    For an end-user:                                               */
/*                                                                   */
/*      1. It allows an end user to set his/her DCE password         */
/*         (encrypted) in his/her RACF profile USRDATA field.        */
/*         To ensure that the password known to the DCE security     */
/*         server is the same, the password is also updated in       */
/*         the DCE registry. If the password has never been set,     */
/*         only the RACF USRDATA field is updated.                   */
/*                                                                   */
/*      2. It allows an end user to login to DCE using the           */
/*         DCE principal name and the encrypted DCE password in his  */
/*         RACF user profile USRDATA. It can be used in a TSO clist  */
/*         executed automatically when a user logon to TSO and thus  */
/*         giving an effective single network logon.                 */
/*                                                                   */
/*    For an administrator:                                          */
/*                                                                   */
/*      3. It allows an Administrator to control who can             */
/*         implicitly login to DCE.  The administrator uses this     */
/*         utility to set the associated DCE principal name in       */
/*         the USRDATA field of the nominated RACF user profile.     */
/*                                                                   */
/*         The administrator must have ALTER access to a RACF        */
/*         general resource 'ADMINISTRATOR' in class '$DECRACF'.     */
/*                                                                   */
/*      4. It allows an Administrator to control who can             */
/*         implicitly logon to RACF. The administrator uses this     */
/*         utility to create a "cross reference UUID" profile and    */
/*         set the RACF userid associated with a DCE principal       */
/*                                                                   */
/*         The administrator must have ALTER access to a RACF        */
/*         general resource 'ADMINISTRATOR' in class '$DECRACF'.     */
/*                                                                   */
/*      5. It allows an Administrator to list the DCE segment        */
/*         stored in the USRDATA field of the nominated RACF         */
/*         userid.                                                   */
/*                                                                   */
/*         The administrator must have ALTER access to a RACF        */
/*         general resource 'ADMINISTRATOR' in class '$DECRACF'.     */
/*                                                                   */
/*      6. It allows an Administrator to list the "cross reference   */
/*         UUID" profile for a DCE principal.                        */
/*                                                                   */
/*         The administrator must have ALTER access to a RACF        */
/*         general resource 'ADMINISTRATOR' in class '$DECRACF'.     */
/*                                                                   */
/*  ENTRY POINT:                                                     */
/*                                                                   */
/*        DCERACFU                                                   */
/*                                                                   */
/*  ENTERED FROM:                                                    */
/*                                                                   */
/*        MVS initiator, TSO call command, OpenEdition Shell         */
/*                                                                   */
/*  PARAMETERS/SYNTAX                                                */
```

Figure 18 (Part 1 of 13). C Routine, DCERACFU

```
/*                                                                    */
/*       The parameters are passed on the command line or in          */
/*       PARM field of the JCL as follows:                            */
/*                                                                    */
/*       'dcelogin'                                                   */
/*                                                                    */
/*       'set password   <password> <verify password>'                */
/*                                                                    */
/*       'set racf       <DCE principal name> <RACF userid>'          */
/*                                                                    */
/*       'set dce        <DCE principal name> <RACF userid>'          */
/*                                                                    */
/*       'list racf      <RACF userid> <-c|-h>'                       */
/*                                                                    */
/*       'list dce       <DCE principal name>'                        */
/*                                                                    */
/*  EXTERNAL REFERENCES: None                                         */
/*              ROUTINES: DCERACF assembler routine                   */
/*                                                                    */
/*  EXITS: NORMAL--                                                   */
/*         ERROR---                                                   */
/*                                                                    */
/*  MACROS/INCLUDES USED:                                             */
/*                                                                    */
/*        DCERACF.H                                                   */
/*                                                                    */
/*  ATTRIBUTES: Problem state, user key                               */
/*                                                                    */
/*  NOTES:                                                            */
/*                                                                    */
/*     This code uses the C tri graph characters for:                 */
/*                                                                    */
/*         left  square bracket   ??(                                 */
/*         right square bracket   ??)                                 */
/*                                                                    */
/*     to avoid possible character set complications on 3270          */
/*     terminals/emulators.                                           */
/*                                                                    */
/*  CHANGE ACTIVITY:                                                  */
/*                                                                    */
/*  04 Jan 95 Module created                       Eric Finkelstein   */
/*                                                                    */
/**********************************************************************/

/* ------------------------ */
/* pragmas                  */
/* ------------------------ */

#pragma linkage(DCERACF,OS)
#pragma runopts(POSIX(ON))

/* ------------------------ */
/* included headers         */
/* ------------------------ */

#include <string.h>
#include <stdio.h>
#include <pthread.h>
#include <dce/rpc.h>
#include <dce/uuid.h>
#include <dce/sec_login.h>
#include <dce/keymgmt.h>
#include <dce/binding.h>
#include <dce/acct.h>
#include "dceracf.h"

/* ------------------------ */
/* defined constants        */
/* ------------------------ */
```

Figure 18 (Part 2 of 13). C Routine, DCERACFU

```
#define nul     0x0
#define DEBUG   0

/* ------------------------ */
/* prototypes               */
/* ------------------------ */

void set_password();
void set_racf();
void set_dce();
void list_dce();
void list_racf();
int  dcelogin();
void convert_principal_to_uuid_string(
        char *principal,
        char *uuid_string);
void check_DCE_status(
        error_status_t   status,
        char             *msg);

/* ------------------------ */
/* global variables         */
/* ------------------------ */

unsigned32   rc;
unsigned32   bh;
unsigned32   i;
unsigned32   command_found;
boolean32    identity_valid;
boolean32    reset_passwd;
char         command[10];
char         subcommand[15];
char         userid[9];
char         principal[129];
char         usrdata[257];
char         password[129];
char         old_password[129];
char         new_password[129];
char         verify_password[129];
char         msg[129];
char         format[20];
char         registry_site[256];
char         dce_error_string[256];
char         uuid_string[40];
sec_login_handle_t          login_context;
sec_rgy_handle_t            registry_handle;
error_status_t              error_inq_st;
sec_passwd_rec_t            pwrec;
sec_passwd_rec_t            caller_key;
sec_passwd_version_t        new_password_version;
sec_login_auth_src_t        auth_src;
sec_rgy_bind_auth_info_t    registry_auth_info;
sec_rgy_login_name_t        login_name;

/* ------------------------ */
/* Main                     */
/* ------------------------ */

int main (int argc, char *argv[])
{

   command_found=0;

   /* ---------------------- */
   /* get command            */
   /* ---------------------- */

   strcpy(command, argv[1]);
   for (i=0; i <= strlen(command); i++)
   {
```

Figure 18 (Part 3 of 13). C Routine, DCERACFU

```
   command[i] = tolower(command[i]);
}

/* ---------------------- */
/* set command            */
/* ---------------------- */

if (strcmp(command, "set") == 0 )
{

   /* ---------------------- */
   /* get subcommand         */
   /* ---------------------- */

   strcpy(subcommand, argv[2]);
   for (i=0; i <= strlen(subcommand); i++)
   {
      subcommand[i] = tolower(subcommand[i]);
   }

   /* ---------------------- */
   /* set password           */
   /* ---------------------- */

   if (strcmp(subcommand, "password") == 0)
   {
      command_found = 1;
      strcpy(new_password, argv[3]);
      strcpy(verify_password, argv[4]);
      set_password();
   }

   /* ---------------------- */
   /* set racf               */
   /* ---------------------- */

   if (strcmp(subcommand, "racf") == 0)
   {
      command_found = 1;
      strcpy(principal, argv[3]);
      strcpy(userid, argv[4]);
      set_racf();
   }

   /* ---------------------- */
   /* set dce                */
   /* ---------------------- */

   if (strcmp(subcommand, "dce") == 0)
   {
      command_found = 1;
      strcpy(principal, argv[3]);
      strcpy(userid, argv[4]);
      set_dce();
   }
}

/* ---------------------- */
/* list                   */
/* ---------------------- */

if (strcmp(command, "list") == 0)
{

   /* ---------------------- */
   /* get subcommand         */
   /* ---------------------- */

   strcpy(subcommand, argv[2]);
   for (i=0; i <= strlen(subcommand); i++)
```

Figure 18 (Part 4 of 13). C Routine, DCERACFU

```
   {
      subcommand[i] = tolower(subcommand[i]);
   }

   /* -------------------- */
   /* list dce             */
   /* -------------------- */

   if (strcmp(subcommand, "dce") == 0)
   {
      command_found = 1;
      strcpy(principal, argv[3]);
      list_dce();
   }

   /* ------------------------ */
   /* list racf                */
   /* ------------------------ */

   if (strcmp(subcommand, "racf") == 0)
   {
      command_found = 1;
      strcpy(userid, argv[3]);
      strcpy(format,"C");
      if (argc == 5)
      {
         strcpy(format, argv[4]);
      }
      list_racf();
   }
}

/* ---------------------- */
/* dcelogin               */
/* ---------------------- */

if (strcmp(command, "dcelogin") == 0)
{
   command_found = 1;
   rc = dcelogin();
   if (rc == 0)
   {
      printf("DCE Login successful for principal=%s\n", principal);
      fflush (NULL);
   }
}

/* ---------------------- */
/* command not recognised */
/* ---------------------- */

if (command_found == 0)
{
   printf(" \n");
   printf("DCERACFU Syntax:\n");
   printf(" \n");
   printf("DCERACFU _,_ dcelogin __.\n");
   printf("           |\n");
   printf("           |_ set password <password> <verify password> __.\n");
   printf("           |\n");
   printf("           |_ set racf <DCE principal name> <RACF userid> __.\n");
   printf("           |\n");
   printf("           |_ set dce <DCE principal name> <RACF userid> __.\n");
   printf("           |\n");
   printf("           |_ list racf <RACF userid> <-c|-h>__.\n");
   printf("           |\n");
   printf("           |_ list dce <DCE principal name> __.\n");
   printf(" \n");
}
fflush(NULL);
```

Figure 18 (Part 5 of 13). C Routine, DCERACFU

```
}

/* ------------------------------------------------------------------ */
/*                                                                    */
/* Function: dcelogin()                                               */
/*                                                                    */
/* Description: Logon to DCE using the DCE principal name and         */
/*     the encrypted DCE password in the current RACF userid.         */
/*                                                                    */
/* ------------------------------------------------------------------ */

int dcelogin()
{

   /* ---------------------- */
   /* get DCE principal name */
   /* ---------------------- */

   rc = DCERACF(RACF_get_DCE_principal, principal);

   if (rc != 0)
   {
      RACF_get_DCE_principal_error(rc, msg);
      printf("%s, rc=%d\n", msg, rc);
      exit(rc);
   }

   /* ---------------------- */
   /* get DCE password       */
   /* ---------------------- */

   rc = DCERACF(RACF_get_DCE_password, password);

   if (rc != 0)
   {
      RACF_get_DCE_password_error(rc, msg);
      printf("%s, rc=%d\n", msg, rc);
      return(rc);
   }

   if (strlen(password) == 0)
   {
      rc = 1;
      return(rc);
   }

   /* -------------------------------- */
   /* setup identity in a login context */
   /* -------------------------------- */

   sec_login_setup_identity(
      principal,
      sec_login_no_flags,
      &login_context,
      &rc);

   check_DCE_status(rc,
      "Setting identity");

   /* ------------------------ */
   /* setup password record    */
   /* ------------------------ */

   pwrec.key.tagged_union.plain = password;
   pwrec.key.key_type = sec_passwd_plain;
   pwrec.pepper = NULL;
   pwrec.version_number = sec_passwd_c_version_none;

   /* ---------------------- */
   /* Validate the Identity  */
```

Figure 18 (Part 6 of 13). C Routine, DCERACFU

```
   /* ---------------------- */

   identity_valid = sec_login_validate_identity(
         login_context,
         &pwrec,
         &reset_passwd,
         &auth_src,
         &rc);

   check_DCE_status(rc,
      "Validating identity");

   /* ----------------- */
   /* Certify identity  */
   /* ----------------- */

   sec_login_certify_identity(
      login_context,
      &rc);

   check_DCE_status(rc,
      "Certifying identity");


   /* ----------------- */
   /* Set login context */
   /* ----------------- */

   sec_login_set_context(
      login_context,
      &rc);

   check_DCE_status(rc,
      "Setting context");

   /* ------------------------------ */
   /* check login OK and authenicated */
   /* ------------------------------ */

   if (identity_valid &&
      (auth_src == sec_login_auth_src_network))
   {
     rc = 0;
   }
   else
   {
     printf("Failed to login...\n");
     fflush (NULL);
     exit(1);
   }

   return(rc);
}

/* ------------------------------------------------------------------ */
/*                                                                    */
/* Function:  set_password()                                          */
/*                                                                    */
/* Description:                                                       */
/*                                                                    */
/*                                                                    */
/* ------------------------------------------------------------------ */

void set_password()
{

   if (strcmp(new_password,verify_password) != 0)
   {
      printf("Password and verify password are different\n");
   }
```

Figure 18 (Part 7 of 13). C Routine, DCERACFU

```
else
{

   /* ---------------------- */
   /* get DCE password */
   /* ---------------------- */

   strcpy(password, "");
   rc = DCERACF(RACF_get_DCE_password, password);

   /* ------------------------ */
   /* Attempt to logon onto DCE */
   /* ------------------------ */

   if (strlen(password) != 0)
   {

      strcpy(old_password, password);         /* save password */

      rc = dcelogin();

      /* ---------------------------- */
      /* open an update registry site        */
      /* ---------------------------- */

      sec_rgy_site_open_update(
         registry_site,                /* find local registry */
         &registry_handle,              /* registry context    */
         &rc);                          /* return code         */

      check_DCE_status(rc,
         "Binding to registry");

      /* ---------------------------- */
      /* use principal set by dcelogin */
      /* ---------------------------- */

      strcpy(login_name.pname, principal);
      strcpy(login_name.gname, "");
      strcpy(login_name.oname, "");

      /* --------------------------------- */
      /* create caller key password record  */
      /* --------------------------------- */

      caller_key.key.tagged_union.plain = old_password;
      caller_key.key.key_type = sec_passwd_plain;
      caller_key.pepper = NULL;
      caller_key.version_number = sec_passwd_c_version_none;

      /* ---------------------------- */
      /* create new password record    */
      /* ---------------------------- */

      pwrec.key.tagged_union.plain = new_password;
      pwrec.key.key_type = sec_passwd_plain;
      pwrec.pepper = NULL;
      pwrec.version_number = sec_passwd_c_version_none;

      /* ---------------------------- */
      /* change the password           */
      /* ---------------------------- */

      sec_rgy_acct_passwd(
         registry_handle,
         &login_name,
         &caller_key,
         &pwrec,
         sec_passwd_des,
         &new_password_version,
```

Figure 18 (Part 8 of 13). C Routine, DCERACFU

```
            &rc);

         check_DCE_status(rc,
            "Changing password in the registry");

         /* ------------------------------ */
         /* unbind registry site                */
         /* ------------------------------ */

         sec_rgy_site_close(registry_handle, &rc);

         check_DCE_status(rc,
            "Closing registry");

      }

      rc = DCERACF(RACF_set_DCE_password, new_password);

      if (rc == 0)
      {
         printf("DCE password has been set in RACF user profile\n");
      }
      else
      {
         RACF_set_DCE_password_error(rc, msg);
         printf("%s, rc=%d\n", msg, rc);
      }
   }
}

/* ------------------------------------------------------------------ */
/*                                                                    */
/* Function: set_racf()                                               */
/*                                                                    */
/* Description: Set the DCE principal name in the RACF user profile  */
/*                                                                    */
/* ------------------------------------------------------------------ */

void set_racf()
{
   for (i=0; i <= strlen(userid); i++)
      userid[i] = toupper(userid[i]);

   rc = DCERACF(RACF_set_DCE_principal, userid, principal);

   if (rc == 0)
   {
      printf("DCE principal=%s, has been set in RACF user profile=%s\n",
              principal, userid);
   }
   else
   {
      RACF_set_DCE_principal_error(rc, msg);
      printf("%s, rc=%d\n", msg, rc);
   }
}

/* ------------------------------------------------------------------ */
/*                                                                    */
/* Function: set_dce()                                                */
/*                                                                    */
/* Description: Create and set RACF userid in a $DECRACF class        */
/*     UUID profile in RACF database                                  */
/*                                                                    */
/* ------------------------------------------------------------------ */

void set_dce()
{
   for (i=0; i <= strlen(userid); i++)
   {
```

Figure 18 (Part 9 of 13). C Routine, DCERACFU

```
      userid[i] = toupper(userid[i]);
   }

   convert_principal_to_uuid_string(principal, uuid_string);

   rc = DCERACF(RACF_set_DCE_uuid, uuid_string, userid);

   if (rc == 0)
   {
      printf("DCE principal=%s, RACF userid=%s, UUID=%s, has been set\n",
              principal, userid, uuid_string);
   }
   else
   {
      RACF_set_DCE_uuid_error(rc, msg);
      printf("%s, rc=%d\n", msg, rc);
   }
}

/* ------------------------------------------------------------------ */
/*                                                                    */
/* Function: list_dce()                                               */
/*                                                                    */
/* Description: Convert principal name to its UUID and list the       */
/*     contents of the UUID profile in RACF database.                 */
/*                                                                    */
/* ------------------------------------------------------------------ */

void list_dce()
{

   convert_principal_to_uuid_string(principal, uuid_string);

   rc = DCERACF(RACF_get_DCE_uuid, uuid_string, userid);

   if (rc == 0)
   {
      printf("DCE principal=%s, RACF userid=%s, UUID=%s\n",
              principal, userid, uuid_string);
   }
   else
   {
      RACF_get_DCE_uuid_error(rc, msg);
      printf("%s, rc=%d\n", msg, rc);
   }
}

/* ------------------------------------------------------------------ */
/*                                                                    */
/* Function: convert_principal_to_uuid_string()                               */
/*                                                                    */
/* Description: Convert principal name to its UUID                    */
/*                                                                    */
/* ------------------------------------------------------------------ */

void convert_principal_to_uuid_string(
        char *principal,
        char *uuid_string)
{
   /* ------------------------ */
   /* local variables          */
   /* ------------------------ */

   int                  i;
   int                  rc;
   int                  userid_found;
   uuid_t               uuid;
   rpc_authz_handle_t   Credentials;
   sec_id_pac_t         *pac;
   unsigned32           authz_svc;
```

Figure 18 (Part 10 of 13). C Routine, DCERACFU

```
unsigned32           status;
char                 *converted_uuid;

/* ---------------------- */
/* initialise output      */
/* ---------------------- */

strcpy(uuid_string, "");
rc = 0;

/* ---------------------------- */
/* open an query registry site   */
/* ---------------------------- */

strcpy(registry_site, "");
sec_rgy_site_open_query(
   registry_site,                    /* find local registry */
   &registry_handle,                 /* registry context    */
   &rc);                             /* return code         */

check_DCE_status(rc,
   "Opening registry");

/* ------------------ */
/* set principal name */
/* ------------------ */

strcpy(login_name.pname, principal);
strcpy(login_name.gname, "");
strcpy(login_name.oname, "");

/* ---------------------- */
/* convert name to uuid   */
/* ---------------------- */

sec_rgy_pgo_name_to_id(
   registry_handle,
   sec_rgy_domain_person,
   &login_name,
   &uuid,
   &rc);

check_DCE_status(rc,
   "convert name to uuid");

/* ---------------------------- */
/* unbind registry site                      */
/* ---------------------------- */

sec_rgy_site_close(registry_handle, &rc);

check_DCE_status(rc,
   "Closing registry");

/* ---------------------- */
/* convert principal UUID */
/* ---------------------- */

uuid_to_string(
   &uuid,
   &converted_uuid,
   &rc );

check_DCE_status(rc,
   "Convert UUID to string");

for (i=0; i <= strlen(converted_uuid); i++)
{
   converted_uuid[i] = toupper(converted_uuid[i]);
}
```

Figure 18 (Part 11 of 13). C Routine, DCERACFU

```
   strcpy(uuid_string, converted_uuid);

   /* ---------------------- */
   /* free UUID string       */
   /* ---------------------- */

   rpc_string_free(
      &converted_uuid,
      &status );

   check_DCE_status(status,
      "Freeing uuid string");

}

/* ---------------------------------------------------------------- */
/*                                                                  */
/* Function: list_racf()                                            */
/*                                                                  */
/* Description: list the usrdata field of the RACF userid           */
/*                                                                  */
/* ---------------------------------------------------------------- */

void list_racf()
{
   for (i=0; i <= strlen(userid); i++)
   {
      userid[i] = toupper(userid[i]);
   }

   for (i=0; i <= strlen(format); i++)
   {
      format[i] = toupper(format[i]);
   }

   if ((strcmp(format, "-C") == 0) ||
       (strcmp(format, "C") == 0))
   {
      strcpy(format, "C");
   }

   rc = DCERACF(RACF_get_DCE_usrdata, userid, format, usrdata);

   if (rc == 0)
   {
      printf("RACF userid=%s, DCE segment=%s\n", userid, usrdata);
   }
   else
   {
      RACF_get_DCE_usrdata_error(rc, msg);
      printf("%s, rc=%d\n", msg, rc);
   }
}

/*********************************************************************/
/*                                                                   */
/* Name:        check_DCE_status                                     */
/*                                                                   */
/* Description: check_DCE_status() implements status checking routine.*/
/*                                                                   */
/*********************************************************************/

void  check_DCE_status(
   error_status_t   status,
   char             *msg)
{

   /* ------------------------ */
   /* local variables          */
```

Figure 18 (Part 12 of 13). C Routine, DCERACFU

```
   /* ------------------------ */

   char              dce_error_string[256];
   error_status_t    error_inq_st;

   /* ------------------------ */
   /* test status              */
   /* ------------------------ */

   if ( status == error_status_ok )
   {
      if (DEBUG == 1)
      {
         printf("%s - OK\n", msg);
      }
   }
   else
   {
      printf("%s - ERROR\n", msg);

      dce_error_inq_text(
         status,
         dce_error_string,
         &error_inq_st);

      printf( "%s\n", dce_error_string);
      fflush(NULL);

      exit ( status );
   }
}
```

Figure 18 (Part 13 of 13). C Routine, DCERACFU

B.3.2 GETPAC

```
/*********************************************************************/
/*                                                                   */
/* MODULE NAME: GETPAC                                               */
/*                                                                   */
/* DESCRIPTIVE NAME: Get DCE principal UUID from the PAC             */
/*                                                                   */
/* FUNCTION:                                                         */
/*                                                                   */
/*    Get DCE principal UUID  from the Caller's DCE                  */
/*    binding handle and PAC (Privilege Attribute Certificate) and   */
/*    return it to the DCERACF user SVC.                             */
/*                                                                   */
/* PROCESSOR: C                                                      */
/*                                                                   */
/* ENTRY POINT:                                                      */
/*                                                                   */
/*         GETPAC                                                    */
/*                                                                   */
/* ENTERED FROM:                                                     */
/*                                                                   */
/*         DCE user SVC                                              */
/*         PASSTICK                                                  */
/*                                                                   */
/* PARAMETERS                                                        */
/*                                                                   */
/*        bh        -  binding handle (input)                        */
/*                                                                   */
/*        UUID      -  principal UUID string  (output)               */
/*                     (null terminated)                             */
/*                                                                   */
/*  EXTERNAL REFERENCES:                                             */
/*             ROUTINES:                                             */
/*                                                                   */
/*  EXITS: NORMAL--                                                  */
/*         ERROR---                                                  */
/*                                                                   */
/*  MACROS/INCLUDES USED:                                            */
/*                                                                   */
/*  ATTRIBUTES:                                                      */
/*                                                                   */
/*  NOTES:                                                           */
/*                                                                   */
/*     This code uses the C tri graph characters for:                */
/*                                                                   */
/*         left  square bracket   ??(                                */
/*         right square bracket   ??)                                */
/*                                                                   */
/*     to avoid possible character set complications on 3270         */
/*     terminals/emulators.                                          */
/*                                                                   */
/*  CHANGE ACTIVITY:                                                 */
/*                                                                   */
/*  19/1/93 Module created                          Eric Finkelstein */
/*                                                                   */
/*********************************************************************/

/* ------------------------ */
/* pragmas                  */
/* ------------------------ */

#pragma linkage(GETPAC,OS)

/* ------------------------ */
/* included headers         */
/* ------------------------ */
```

Figure 19 (Part 1 of 4). C Routine, GETPAC

```
#include <string.h>
#include <stdio.h>
#include <pthread.h>
#include <dce/rpc.h>
#include <dce/uuid.h>
#include <dce/sec_login.h>
#include <dce/keymgmt.h>
#include <dce/binding.h>
#include <dce/acct.h>

/* ------------------------ */
/* defined constants        */
/* ------------------------ */

#define DEBUG               0
#define OK                  0
#define NO_PAC              12
#define NO_RACF_USERID      16

/* ------------------------ */
/* prototypes               */
/* ------------------------ */

void  getpac_check_DCE_status(
   error_status_t   status,
   char             *msg);

/* ------------------------ */
/* global variables         */
/* ------------------------ */

/* ------------------------ */
/* GETPAC                   */
/* ------------------------ */

int GETPAC(
      handle_t    bh,
      char       *principal_uuid)
{

   /* ------------------------ */
   /* local variables          */
   /* ------------------------ */

   int                  i;
   int                  rc;
   int                  userid_found;
   rpc_authz_handle_t   Credentials;
   sec_id_pac_t         *pac;
   unsigned32           authz_svc;
   unsigned32           status;
   unsigned_char_t      *uuid_string;
   char                 group_prefix[25];

   /* ---------------------- */
   /* initialise output      */
   /* ---------------------- */

   strcpy(principal_uuid, "");
   rc = 0;

   /* ---------------------- */
   /* get clients auth info, */
   /* ---------------------- */

   rpc_binding_inq_auth_client(
      bh,                         /* binding handle               */
      &Credentials,               /* returned privileges          */
      NULL,                       /* we provide no principal name  */
```

Figure 19 (Part 2 of 4). C Routine, GETPAC

```
      NULL,                        /* no protection level returned  */
      NULL,                        /* no authn_svc returned         */
      &authz_svc,                  /* Credential contents indicator */
      &status
   );

   getpac_check_DCE_status(status,
      "Inquire clients auth info");

   /* -------------------------------- */
   /* getpac_check the contents of credentials */
   /* -------------------------------- */

   if ( authz_svc != rpc_c_authz_dce )
   {
      rc = NO_PAC;
      return(rc);
   }

   pac = (sec_id_pac_t *) Credentials;

   /* ---------------------- */
   /* convert principal UUID */
   /* ---------------------- */

   uuid_to_string(
      &pac->principal.uuid,
      &uuid_string,
      &status );

   getpac_check_DCE_status(status,
      "Convert UUID to string");

   for (i=0; i <= strlen(uuid_string); i++)
   {
      uuid_string[i] = toupper(uuid_string[i]);
   }

   strcpy(principal_uuid, uuid_string);

   /* ---------------------- */
   /* free UUID string       */
   /* ---------------------- */

   rpc_string_free(
      &uuid_string,
      &status );

   getpac_check_DCE_status(status,
      "Freeing uuid string");

   /* --------- */
   /* return    */
   /* --------- */

   return(rc);

}

/*************************************************************************/
/*                                                                       */
/* Name:        getpac_check_DCE_status                                  */
/*                                                                       */
/* Description: getpac_check_DCE_status() implements status getpac_checking routine.*/
/*                                                                       */
/*************************************************************************/

void  getpac_check_DCE_status(
   error_status_t   status,
   char             *msg)
```

Figure 19 (Part 3 of 4). C Routine, GETPAC

```
{

   /* ------------------------ */
   /* local variables          */
   /* ------------------------ */

   char              dce_error_string[256];
   error_status_t    error_inq_st;

   /* ------------------------ */
   /* test status              */
   /* ------------------------ */

   if ( status == error_status_ok )
   {
      if (DEBUG == 1)
      {
         printf("%s - OK\n", msg);
      }
   }
   else
   {
      printf("%s - ERROR\n", msg);

      dce_error_inq_text(
         status,
         dce_error_string,
         &error_inq_st);

      printf( "%s\n", dce_error_string);
      fflush(NULL);

      exit ( status );
   }
}
```

Figure 19 (Part 4 of 4). C Routine, GETPAC

B.3.3 PTKTLIB

```
/*********************************************************************/
/*                                                                   */
/* MODULE NAME: PTKTLIB                                              */
/*                                                                   */
/* DESCRIPTIVE NAME: Routines for a RACF passticket client           */
/*                                                                   */
/* FUNCTION:                                                         */
/*                                                                   */
/*     This module provides a function to bind a client application  */
/*     to the RACF Passticket server, PTKTSERV, and the request for  */
/*     a Passticket:                                                 */
/*                                                                   */
/*         bind_to_passticket_server(&PTKT_bh, <sysid>);             */
/*         rc = passticket(PTKT_bh, ticket);                         */
/*                                                                   */
/*     where                                                         */
/*                                                                   */
/*         PTKT_bh  - is the binding handle (allocated in the        */
/*                    calling routine) returned for the Passticket   */
/*                    session.                                       */
/*                                                                   */
/*         <sysid>  - is the MVS system id on which both the         */
/*                    Passticket and the target application server   */
/*                    reside.                                        */
/*                                                                   */
/*         ticket   - is the ticket returned by the passticket       */
/*                    function, to be used in a subsequent call      */
/*                    to the target application server.              */
/*                                                                   */
/*  PROCESSOR: C                                                     */
/*                                                                   */
/*  ENTRY POINT:                                                     */
/*                                                                   */
/*         bind_to_passticket_server                                 */
/*         passticket                                                */
/*                                                                   */
/*  ENTERED FROM:                                                    */
/*                                                                   */
/*         'C' DCE clients                                           */
/*                                                                   */
/*  EXTERNAL REFERENCES:                                             */
/*             ROUTINES:                                             */
/*                                                                   */
/*  EXITS: NORMAL--                                                  */
/*         ERROR---                                                  */
/*                                                                   */
/*  MACROS/INCLUDES USED:                                            */
/*                                                                   */
/*  ATTRIBUTES:                                                      */
/*                                                                   */
/*  NOTES:                                                           */
/*                                                                   */
/*  CHANGE ACTIVITY:                                                 */
/*                                                                   */
/*  20/2/95 Module created                          Eric Finkelstein */
/*                                                                   */
/*********************************************************************/

/* ------------------------ */
/* defines                  */
/* ------------------------ */

#define PTKT_IF_HANDLE        PTKT_v1_0_c_ifspec
#define PTKT_CDS_DIRECTORY    "/.:/passticket/"
#define PTKT_SERVER_NAME      "passticket"
```

Figure 20 (Part 1 of 4). C Routine, PTKTLIB

```
/* ------------------------ */
/* included headers         */
/* ------------------------ */

#include <string.h>
#include <stdio.h>
#include <dce/acct.h>
#include <dce/binding.h>
#include <dce/sec_login.h>
#include "ptkt.h"

/* ------------------------ */
/* prototypes               */
/* ------------------------ */

void PTKT_check_DCE_status(
        error_status_t status,
        char *msg);

void bind_to_passticket_server(
        handle_t *PTKT_bh,
        char     *sysid);

/* --------------------------- */
/* GLOBAL variables            */
/* --------------------------- */

int                   PTKT_rc;
handle_t              bh;
char                  PTKT_msg[128];
char                  PTKT_server_entry_name[128];
char                  PTKT_dce_error_string[256];
rpc_ns_handle_t       PTKT_ns_handle;
error_status_t        PTKT_status;
error_status_t        PTKT_error_inq_st;

/*************************************************************/
/*                                                           */
/* Name:        bind_to_passticket_server()                  */
/*                                                           */
/* Description: bind to passticket server                    */
/*                                                           */
/*************************************************************/

void  bind_to_passticket_server(
      handle_t *PTKT_bh,
      char     *sysid)
{
   /* --------------------------- */
   /* build cds server entry name  */
   /* --------------------------- */

   strcpy(PTKT_server_entry_name, PTKT_CDS_DIRECTORY);
   strcat(PTKT_server_entry_name, sysid);

   /* ------------------------------------ */
   /* set up for reading binding information */
   /* ------------------------------------ */

   rpc_ns_binding_import_begin(
      rpc_c_ns_syntax_default,
      PTKT_server_entry_name,
      PTKT_IF_HANDLE,
      NULL,
      &PTKT_ns_handle,
      &PTKT_status
   );

   PTKT_check_DCE_status(PTKT_status,
```

Figure 20 (Part 2 of 4). C Routine, PTKTLIB

```
      "PTKT Binding import begin");

   /* ------------------------------------------------ */
   /* import first binding information from name space */
   /* ------------------------------------------------ */

   rpc_ns_binding_import_next(
      PTKT_ns_handle,
      &bh,
      &PTKT_status
   );

   PTKT_check_DCE_status(PTKT_status,
      "PTKT Import binding");

   /* -------------------------------- */
   /* stop reading binding indormation */
   /* -------------------------------- */

   rpc_ns_binding_import_done(
      &PTKT_ns_handle,
      &PTKT_status
   );

   PTKT_check_DCE_status(PTKT_status,
      "PTKT Import binding done");

   /* ---------------------- */
   /* set authorization info */
   /* ---------------------- */

   rpc_binding_set_auth_info(
      bh,
      PTKT_SERVER_NAME,
      rpc_c_protect_level_pkt_integ,    /* send checksum       */
      rpc_c_authn_default,
      NULL,                             /* use login context   */
      rpc_c_authz_dce,                  /* send PAC            */
      &PTKT_status
   );

   PTKT_check_DCE_status(PTKT_status,
      "PTKT Set authorisation info");

   /* ----------------- */
   /* set caller's bh   */
   /* ----------------- */

   *PTKT_bh = bh;

}

/*****************************************************************/
/*                                                               */
/* Name:        PTKT_check_DCE_status                            */
/*                                                               */
/* Description: status checking routine                          */
/*                                                               */
/*****************************************************************/

void  PTKT_check_DCE_status(
   error_status_t   status,
   char             *msg)
{
   if ( status != error_status_ok )
   {
      printf("PTKT, %s - ERROR\n", msg);

      dce_error_inq_text(
         status,
```

Figure 20 (Part 3 of 4). C Routine, PTKTLIB

```
         PTKT_dce_error_string,
         &PTKT_error_inq_st);

      printf( "%s\n", PTKT_dce_error_string);
      fflush(NULL);

      exit ( status );
   }
}
```

Figure 20 (Part 4 of 4). C Routine, PTKTLIB

B.3.4 PTKTS

```
/*********************************************************************/
/*                                                                   */
/* MODULE NAME: PTKTS                                                */
/*                                                                   */
/* DESCRIPTIVE NAME: A DCE server to generate a RACF passticket.     */
/*                                                                   */
/* FUNCTION:                                                         */
/*                                                                   */
/*     This module is a server module that will generate a RACF      */
/*     passticket for the client. This will allow that client to     */
/*     pass that ticket to a target DCE application server and       */
/*     still unrestricted access to all RACF resources to which      */
/*     he/she has been given access.                                 */
/*                                                                   */
/*     This server must be run in an MVS APF authorized library and  */
/*     and be APF authorized so that it can call the privileged      */
/*     RACF service routine that generates RACF Passtickets.         */
/*                                                                   */
/*     The manager code, "passticket", calls a 'C' interface         */
/*     routine, PTKTGEN, which builds the DCE application name       */
/*     used by the user SVC, "DCE<smf system id>" and calls the      */
/*     RACF passticket generator for it. The SMF system id is set    */
/*     by the installation in SID parameter in member IEASMF0O of    */
/*     'SYS1.PARMLIB'.  Any special characters in that SID are       */
/*     removed. For example, SID=S*Y5 would be used to build         */
/*     'DCESY5' application name,                                    */
/*                                                                   */
/*     The server advertises its presence in the CDS by an entry     */
/*     of the <sysid> in the '/.:/passticket/' directory.            */
/*                                                                   */
/*     For example                                                   */
/*                                                                   */
/*             '/.:/passticket/SY1'                                  */
/*                                                                   */
/*     where the <sysid> is system name in the MVS CVT field,        */
/*     CVTSNAME. This is set by the installation in 'SYS1.PARMLIB'   */
/*     IEASYS0O member, SYSNAME parameter.                           */
/*                                                                   */
/*  PROCESSOR: C                                                     */
/*                                                                   */
/*  ENTRY POINT:                                                     */
/*                                                                   */
/*         PTKTS                                                     */
/*                                                                   */
/*  ENTERED FROM:                                                    */
/*                                                                   */
/*         MVS initiator, TSO CALL command, OpenEdition Shell        */
/*                                                                   */
/*  PARAMETERS:                                                      */
/*                                                                   */
/*         none                                                      */
/*                                                                   */
/*  EXTERNAL REFERENCES:                                             */
/*             ROUTINES:                                             */
/*                                                                   */
/*  EXITS: NORMAL--                                                  */
/*         ERROR---                                                  */
/*                                                                   */
/*  MACROS/INCLUDES USED:                                            */
/*                                                                   */
/*  ATTRIBUTES:                                                      */
/*                                                                   */
/*         APF authorized                                            */
/*                                                                   */
/*  NOTES:                                                           */
```

Figure 21 (Part 1 of 11). C Routine, PTKTS

```
/*                                                                    */
/*  1. This server issues a DCE login for the DCE principal name      */
/*     "passticket".                                                  */
/*                                                                    */
/*  2. A keytab file must be created for "passticket".                */
/*                                                                    */
/*  3. "passticket" must be a member of:                              */
/*         "subsys/dce/rpc-server-group".                             */
/*                                                                    */
/*  4. "passticket" must have:                                        */
/*                                                                    */
/*        "rwdtci" authority for "/.:/passticket"                     */
/*        "rwdtc"  authority for "/.:/passticket/<sysid>"             */
/*                                                                    */
/*  5. This code uses the C tri graph characters for:                 */
/*                                                                    */
/*         left  square bracket   ??(                                 */
/*         right square bracket   ??)                                 */
/*                                                                    */
/*     to avoid possible character set complications on 3270          */
/*     terminals/emulators.                                           */
/*                                                                    */
/*                                                                    */
/*  CHANGE ACTIVITY:                                                  */
/*                                                                    */
/*  18/2/95 Module created                          Eric Finkelstein  */
/*                                                                    */
/**********************************************************************/

/* ---------------------- */
/* pragmas                */
/* ---------------------- */

#pragma linkage(DCERACF,OS)
#pragma linkage(PTKTGEN,OS)
#pragma runopts(posix(on))

/* -------------------------------------- */
/*  included headers                      */
/* -------------------------------------- */

#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/time.h>
#include <pthread.h>
#include <exc@hand.h>
#include <dce/rpc.h>
#include <dce/sec_login.h>
#include <dce/keymgmt.h>
#include <dce/binding.h>
#include <dce/acct.h>
#include "dceracf.h"
#include "ptkt.h"

/* -------------------------------------- */
/*  defined constants                     */
/* -------------------------------------- */

#define MAX_CONC_CALLS_PROTSEQ      2
#define MAX_CONC_CALLS_TOTAL        4
#define IF_HANDLE                   PTKT_v1_0_s_ifspec
#define CDS_DIRECTORY               "/.:/passticket/"
#define SERVER_PRINCIPAL_NAME       "passticket"
#define DEBUG 0

/* -------------------------------------- */
/*  prototypes                            */
/* -------------------------------------- */
```

Figure 21 (Part 2 of 11). C Routine, PTKTS

```
void  check_DCE_status(error_status_t status, char *msg);
void  refresh_login_context_rtn();

/* -------------------- */
/*  Global variables    */
/* -------------------- */

char    keytab[128];

/* ---------------------------------------- */
/*  Main                                    */
/* ---------------------------------------- */

int main (int argc, char *argv[])
{

   /* -------------------------------- */
   /* local variables                  */
   /* -------------------------------- */

   unsigned_char_t       server_entry_name[129];
   unsigned_char_t       uuid_string[129];
   rpc_binding_vector_t  *bind_vector_p;
   sec_login_handle_t    login_context;
   sec_login_auth_src_t  auth_src;
   pthread_t             refresh_login_context_thread;
   void                  *server_key;
   boolean32             identity_valid;
   boolean32             reset_passwd;
   error_status_t        status;
   char                  sysid[9];
   char                  msg[128];
   idl_long_int          rc;

   /* -------------------------------- */
   /* setup identity in a login context */
   /* -------------------------------- */

   sec_login_setup_identity(
      SERVER_PRINCIPAL_NAME,
      sec_login_no_flags,
      &login_context,
      &status);

   check_DCE_status(status,
      "Setting Identity");

   /* ------------------------ */
   /* Get Key From KEYTAB File */
   /* ------------------------ */

   if (argc != 2)
   {
      printf("Keytab file not entered as a parameter\n");
      exit(1);
   }

   strcpy(keytab, argv[1]);
   printf("Keytab=%s\n", keytab);

   sec_key_mgmt_get_key(
      rpc_c_authn_dce_secret,
      keytab,
      SERVER_PRINCIPAL_NAME,
      0,
      &server_key,
      &status);

   check_DCE_status(status,
      "Getting Key from Keytab file");
```

Figure 21 (Part 3 of 11). C Routine, PTKTS

```
/* ---------------------- */
/* Validate the Identity  */
/* ---------------------- */

identity_valid =
   sec_login_validate_identity(
      login_context,
      server_key,
      &reset_passwd,
      &auth_src,
      &status);

check_DCE_status(status,
   "Validate Identity");

/* ----------------- */
/* Certify identity  */
/* ----------------- */

sec_login_certify_identity(
   login_context,
   &status);

check_DCE_status(status,
   "Certifying identity");

/* ----------------- */
/* Set login context */
/* ----------------- */

sec_login_set_context(
   login_context,
   &status);

check_DCE_status(status,
   "Setting context");

/* ---------------- */
/* Free Key Storage */
/* ---------------- */

sec_key_mgmt_free_key(
   &server_key,
   &status);

check_DCE_status(status,
   "Freeing key storage");

/* ------------------------------- */
/* check login OK and authenicated */
/* ------------------------------- */

if (identity_valid &&
   (auth_src == sec_login_auth_src_network))
{
  printf("Logged in as principal=%s\n", SERVER_PRINCIPAL_NAME);
  fflush (NULL);
}
else
{
  printf("Failed to login  as principal=%s\n",
     SERVER_PRINCIPAL_NAME);
  fflush (NULL);
  exit(1);
}

/* ---------------------------------------------- */
/* set the authentication level which will be used */
/* ---------------------------------------------- */
```

Figure 21 (Part 4 of 11). C Routine, PTKTS

```
rpc_server_register_auth_info (
   SERVER_PRINCIPAL_NAME,
   rpc_c_authn_dce_secret,
   NULL,
   keytab,
   &status
);

check_DCE_status( status,
   "Setting authentication level");

/* ------------------------------------------------------ */
/* Register interface/epv associations with rpc runtime. */
/* ------------------------------------------------------ */

rpc_server_register_if (
   IF_HANDLE,
   NULL,
   NULL,
   &status );

check_DCE_status( status,
   "Registering interface/epv");

/* ------------------------------------------------------------ */
/* Inform rpc runtime to use all supported protocol sequences. */
/* ------------------------------------------------------------ */

rpc_server_use_all_protseqs (
   MAX_CONC_CALLS_PROTSEQ,
   &status );

check_DCE_status( status,
   "Setting protocol sequences");

/* ------------------------------------------------- */
/* Ask the runtime which binding handle will be used. */
/* ------------------------------------------------- */


rpc_server_inq_bindings(
   &bind_vector_p,
   &status );

check_DCE_status( status,
   "Getting binding handle");

/* ---------------------------------------------- */
/* Register binding information with endpoint map */
/* ---------------------------------------------- */

rpc_ep_register(
   IF_HANDLE,
   bind_vector_p,
   NULL,
   "SAVEPW server, version 1.0",
   &status);

check_DCE_status( status,
   "Registering endpoint");

/* -------------------- */
/* Get <sysid> from CVT */
/* -------------------- */

rc = DCERACF(RACF_get_MVS_sysid, sysid);

/* ---------------------- */
/* build server entry name */
```

Figure 21 (Part 5 of 11). C Routine, PTKTS

```
/* ---------------------- */

strcpy(server_entry_name, CDS_DIRECTORY);
strcat(server_entry_name, sysid);

/* ------------------------------------ */
/* Export binding info to the namespace. */
/* ------------------------------------ */

rpc_ns_binding_export (
   rpc_c_ns_syntax_default,
   server_entry_name,
   IF_HANDLE,
   bind_vector_p,
   NULL,
   &status
);

check_DCE_status( status,
   "Exporting binding information");

/* -------------------------- */
/* Start refresh login thread   */
/* -------------------------- */

rc = pthread_create(
   &refresh_login_context_thread,
   pthread_attr_default,
   (pthread_startroutine_t) refresh_login_context_rtn,
   (pthread_addr_t) login_context);

if (rc == -1)
{
   printf("Failed to create refresh login context thread\n");
   exit(1);
}

/* -------------------------- */
/* Listen for service requests. */
/* -------------------------- */

TRY {

   printf("Server entry name=%s\n",
           server_entry_name);

   printf( "Server listening\n");
   fflush(NULL);

   rpc_server_listen (
      MAX_CONC_CALLS_TOTAL,
      &status );

   check_DCE_status( status,
      "Listening for service requests");

}

/* ---------------------------------------- */
/* shutdown request from an authorised client */
/* ---------------------------------------- */

FINALLY {

   /* ------------------------------------------------- */
   /* Unexport the binding information from the namespace. */
   /* ------------------------------------------------- */

   rpc_ns_binding_unexport (
      rpc_c_ns_syntax_default,
```

Figure 21 (Part 6 of 11). C Routine, PTKTS

```
         server_entry_name,
         IF_HANDLE,
         NULL,
         &status);

      check_DCE_status( status,
         "Unexporting binding information");

      /* ----------------------------------- */
      /* Unregister interface from RPC runtime */
      /* ----------------------------------- */

      rpc_server_unregister_if (
         IF_HANDLE,
         NULL,
         &status );

      check_DCE_status( status,
         "Unregistering interface");

      /* ---------------------------- */
      /* Unregister interface from EPV */
      /* ---------------------------- */

      rpc_ep_unregister (
         IF_HANDLE,
         bind_vector_p,
         NULL,
         &status );

      check_DCE_status( status,
         "Unregistering interface from EPV");

      /* ------------------------------- */
      /* exit                            */
      /* ------------------------------- */

      exit ( 0 );
   }
   ENDTRY;
}

/*********************************************************************/
/*                                                                   */
/* Function:    refresh_login_context_rtn                            */
/*                                                                   */
/* Description: Every hour, refresh login context                    */
/*                                                                   */
/*********************************************************************/

void refresh_login_context_rtn(
        sec_login_handle_t  login_context)
{

   /* ------------------------------ */
   /* local variables                */
   /* ------------------------------ */

   signed32         current_time;
   signed32         expiration;
   signed32         delay_time;
   struct timespec  delay;
   unsigned32       used_kvno;
   boolean32        reset_password;
   boolean32        identity_valid;
   void             *server_key;
   sec_login_auth_src_t  auth_src;
   error_status_t   status;
   #define          MINUTE 60
   #define          DAYSECS 86400
```

Figure 21 (Part 7 of 11). C Routine, PTKTS

```
/* ------------------- */
/* infinite loop       */
/* ------------------- */

while (1)
{

   /* ---------------------------------------- */
   /* wait till logon context is about to expire */
   /* ---------------------------------------- */

   sec_login_get_expiration (
      login_context,
      &expiration,
      &status);

   if ((status != rpc_s_ok) &&
       (status != sec_login_s_not_certified))
   {
      printf("Cannot get login context expiration time\n");
      fflush (NULL);
      exit(1);
   }

   current_time = time(NULL);
   delay_time = expiration - current_time - (10*MINUTE);

   if (delay_time > 0)
   {
      delay.tv_sec = delay_time;
      delay.tv_nsec = 0;
      pthread_delay_np(&delay);
   }

   /* ----------------------------- */
   /* refresh identity              */
   /* ----------------------------- */

   sec_login_refresh_identity(
      login_context,
      &status);

   check_DCE_status(status,
      "Refreshing identity file");

   /* ----------------------- */
   /* Get Key From KEYTAB File */
   /* ----------------------- */

   sec_key_mgmt_get_key(
      rpc_c_authn_dce_secret,
      keytab,
      SERVER_PRINCIPAL_NAME,
      0,
      &server_key,
      &status);

   check_DCE_status(status,
      "Getting Key from Keytab file");

   /* ---------------------- */
   /* Validate the Identity  */
   /* ---------------------- */

   identity_valid =
      sec_login_validate_identity(
         login_context,
         server_key,
         &reset_password,
```

Figure 21 (Part 8 of 11). C Routine, PTKTS

```
            &auth_src,
            &status);

      check_DCE_status(status,
         "Validating Identity");

      /* ---------------- */
      /* Free Key Storage */
      /* ---------------- */

      sec_key_mgmt_free_key(
         &server_key,
         &status);

      check_DCE_status(status,
         "Freeing key storage");

      /* ---------------- */
      /* check refresh OK */
      /* ---------------- */

      if (!identity_valid)
      {
        printf("Refresh of login context failed\n");
        fflush (NULL);
      }
   }
}

/*********************************************************************/
/*                                                                   */
/* Name:        check_DCE_status                                     */
/*                                                                   */
/* Description: check_DCE_status() implements status checking routine.*/
/*                                                                   */
/*********************************************************************/

void  check_DCE_status(error_status_t status, char *msg)
{

   /* ------------------- */
   /* local variables     */
   /* ------------------- */

   char            dce_error_string[256];
   error_status_t  error_inq_st;

   /* ------------------- */
   /* check status        */
   /* ------------------- */

   if ( status == error_status_ok )
   {
      if (DEBUG == 1)
      {
         printf("%s - OK\n", msg);
      }
   }
   else
   {
      printf("%s - ERROR\n", msg);

      dce_error_inq_text(
         status,
         dce_error_string,
         &error_inq_st);

      printf( "%s\n", dce_error_string);
      fflush(NULL);
```

Figure 21 (Part 9 of 11). C Routine, PTKTS

```
      exit ( status );
   }
}

/*********************************************************************/
/*                                                                   */
/* Function:    passticket                                           */
/*                                                                   */
/* Description: generate a RACF passticket                           */
/*                                                                   */
/*********************************************************************/

idl_long_int passticket(handle_t bh, char *ticket)
{

   /* ------------------------------ */
   /* defines                        */
   /* ------------------------------ */

   #define passticket_unable_to_get_uuid_string_no_PAC          1
   #define passticket_unable_to_convert_uuid_to_RACF_userid     2
   #define passticket_unable_to_generate_ticket                 3

   /* ------------------------------ */
   /* local variables                */
   /* ------------------------------ */

   idl_long_int rc;
   char         msg[128];
   char         userid[9];
   char         uuid[37];
   error_status_t   status;

   /* ---------------------------------- */
   /* get principal uuid string from PAC */
   /* ---------------------------------- */

   rc = GETPAC(bh, uuid);

   if (rc != 0)
   {
      strcpy(msg, "Unable to get uuid string");
      printf("%s - rc=%d\n", msg, rc);
      fflush(NULL);
      rc = passticket_unable_to_get_uuid_string_no_PAC;
      return(rc);
   }

   /* --------------------------- */
   /* convert to RACF userid      */
   /* --------------------------- */

   rc = DCERACF(RACF_get_DCE_uuid, uuid, userid);

   if (rc != 0)
   {
      RACF_get_DCE_uuid_error(rc, msg);
      printf("%s - rc=%d\n", msg, rc);
      fflush(NULL);
      rc = passticket_unable_to_convert_uuid_to_RACF_userid;
      return(rc);
   }

   /* --------------------------- */
   /* generate a ticket           */
   /* --------------------------- */

   rc = PTKTGEN(userid, ticket);

   if (rc != 0)
```

Figure 21 (Part 10 of 11). C Routine, PTKTS

```
   {
      strcpy(msg, "Unable to generate ticket");
      printf("%s - rc=%d\n", msg, rc);
      fflush(NULL);
      rc = passticket_unable_to_generate_ticket;
      return(rc);
   }
   printf("ticket created for '%s'\n", userid);
   return(rc);

}
```

Figure 21 (Part 11 of 11). C Routine, PTKTS

B.3.5 PTSTC

```
/*********************************************************************/
/*                                                                   */
/* MODULE NAME: PTSTC                                                */
/*                                                                   */
/* DESCRIPTIVE NAME: Test the RACF passticket generator              */
/*                                                                   */
/* FUNCTION:                                                         */
/*                                                                   */
/*     This module is a client module that is used to test the       */
/*     RACF passticket server, PTKTSERV. The test application        */
/*     adds a single record to a sequential MVS data set. The data   */
/*     name is not known to the application server, and the RACF     */
/*     userid the application server under which it is running does  */
/*     not have access to the data sets. Only the client has access  */
/*     has update access. The data name must be unqualified.         */
/*                                                                   */
/*     This test application is also used to demonstrate the use     */
/*     of the RACF passticket server, PTKTSERV.                      */
/*                                                                   */
/*     The RACF passticket server provides a header file, PTKTLIB.H  */
/*     which contains two functions:                                 */
/*                                                                   */
/*         bind_to_passticket_server(&PTKT_bh, <sysid>);             */
/*         rc = passticket(PTKT_bh, ticket);                         */
/*                                                                   */
/*                                                                   */
/*  PROCESSOR: C                                                     */
/*                                                                   */
/*  ENTRY POINT:                                                     */
/*                                                                   */
/*         PTSTC                                                     */
/*                                                                   */
/*  ENTERED FROM:                                                    */
/*                                                                   */
/*         MVS initiator, TSO CALL command, OpenEdition Shell        */
/*                                                                   */
/*  SYNTAX:                                                          */
/*                                                                   */
/*         PTSTC <sysid> <dsn> <record>                              */
/*                                                                   */
/*  EXTERNAL REFERENCES:                                             */
/*             ROUTINES:                                             */
/*                                                                   */
/*  EXITS: NORMAL--                                                  */
/*         ERROR---                                                  */
/*                                                                   */
/*  MACROS/INCLUDES USED:                                            */
/*                                                                   */
/*  ATTRIBUTES:                                                      */
/*                                                                   */
/*  NOTES:                                                           */
/*                                                                   */
/*  CHANGE ACTIVITY:                                                 */
/*                                                                   */
/*  20/2/95 Module created                        Eric Finkelstein   */
/*                                                                   */
/*********************************************************************/

/* ------------------------ */
/* pragmas                  */
/* ------------------------ */

#pragma runopts(posix(on))

/* ------------------------ */
```

Figure 22 (Part 1 of 5). C Routine, PTSTC

```
/* included headers         */
/* ------------------------ */

#include <string.h>
#include <stdio.h>
#include <dce/acct.h>
#include <dce/binding.h>
#include <dce/sec_login.h>
#include <dce/pgo.h>
#include <dce/uuid.h>
#include "ptktlib.h"
#include "ptst.h"

/* ------------------------ */
/* defined constants        */
/* ------------------------ */

#define IF_HANDLE        PTST_v1_0_c_ifspec
#define CDS_DIRECTORY    "/.:/passticket/ptst/"
#define SERVER_NAME      "ptst"
#define DEBUG            0
#define OK               0

/* ------------------------ */
/* prototypes               */
/* ------------------------ */

void check_DCE_status(error_status_t status, char *msg);
void bind_to_test_server();

/* --------------------- */
/*  message definitions  */
/* --------------------- */

char *ptst_msg[] = {
   /*0*/  "Record added succesfully",
   /*1*/  "Record not added, password or RACF userid invalid",
   /*2*/  "PTST server stopped",
   /*3*/  "You are not authorized to stop the PTST server",
   /*4*/  "Unable to open file",
   /*x*/   ""         };

/* ------------------------ */
/* global variables         */
/* ------------------------ */

int                   i;
idl_long_int          rc;
handle_t              bh;
handle_t              PTKT_bh;
char                  msg[128];
char                  server_entry_name[128];
char                  dce_error_string[256];
char                  sysid[9];
char                  ticket[9];
char                  dsn[255];
char                  record[255];
sec_login_handle_t    *login_context;
rpc_ns_handle_t       ns_handle;
error_status_t        status;
error_status_t        error_inq_st;

/* ------------------------ */
/* MAIN                     */
/* ------------------------ */

int main (int argc, char *argv[])
{

   /* --------------------- */
```

Figure 22 (Part 2 of 5). C Routine, PTSTC

```
/* parse the command line */
/* ---------------------- */

if (argc == 4)
{

   strcpy(sysid, argv[1]);

   for (i=0; i <= strlen(sysid); i++)
   {
      sysid[i] = toupper(sysid[i]);
   }

   strcpy(dsn, argv[2]);

   for (i=0; i <= strlen(dsn); i++)
   {
      dsn[i] = toupper(dsn[i]);
   }

   strcpy(record, argv[3]);

}
else
{
   if (argc == 3)
   {
      strcpy(sysid, argv[1]);

      for (i=0; i <= strlen(sysid); i++)
      {
         sysid[i] = toupper(sysid[i]);
      }

      strcpy(dsn, argv[2]);

      for (i=0; i <= strlen(dsn); i++)
      {
         dsn[i] = toupper(dsn[i]);
      }

      strcpy(record, "STOP");

      if (strcmp(dsn, "STOP") != 0)
      {
         printf("Syntax: PTSTC <sysid> <dsn> <'record'>\n");
         exit(1);
      }
   }
}

/* ------------------------- */
/* bind to passticket server */
/* ------------------------- */

printf("sysid=%s, dsn=%s, record=%s\n", sysid, dsn, record);

bind_to_passticket_server(&PTKT_bh, sysid);

printf("Bound to passticket server\n");

/* ------------------- */
/* bind to test server */
/* ------------------- */

bind_to_test_server();
printf("Bound to test server\n");

/* ----------------------------------- */
/* call the passticket server          */
```

Figure 22 (Part 3 of 5). C Routine, PTSTC

```
   /* ------------------------------------ */

   strcpy(ticket, "");
   printf("Getting ticket\n");
   rc = passticket(PTKT_bh, ticket);
   if (rc != 0)                        /* if error on server */
   {
      printf("Failed to get a ticket\n");
      exit(2);
   }
   printf("Got ticket\n");

   /* ------------------------------------ */
   /* call the test application server     */
   /* ------------------------------------ */

   printf("Calling test server\n");

   rc = ptst(bh, ticket, dsn, record);

   printf("%s\n", ptst_msg[rc]);

   exit ();
}

/*********************************************************************/
/*                                                                   */
/* Name:        bind_to_test_server()                                */
/*                                                                   */
/* Description: bind to test server                                  */
/*                                                                   */
/*********************************************************************/

void  bind_to_test_server()
{
   /* --------------------------- */
   /* build cds server entry name  */
   /* --------------------------- */

   strcpy(server_entry_name, CDS_DIRECTORY);
   strcat(server_entry_name, sysid);

   /* ------------------------------------ */
   /* set up for reading binding information */
   /* ------------------------------------ */

   rpc_ns_binding_import_begin(
      rpc_c_ns_syntax_default,
      server_entry_name,
      IF_HANDLE,
      NULL,
      &ns_handle,
      &status
   );

   check_DCE_status(status,
      "Binding import begin");

   /* ---------------------------------------------- */
   /* import first binding information from name space */
   /* ---------------------------------------------- */

   rpc_ns_binding_import_next(
      ns_handle,
      &bh,
      &status
   );

   check_DCE_status(status,
      "Import binding");
```

Figure 22 (Part 4 of 5). C Routine, PTSTC

```
   /* ------------------------------- */
   /* stop reading binding indormation */
   /* ------------------------------- */

   rpc_ns_binding_import_done(
      &ns_handle,
      &status
   );

   check_DCE_status(status,
      "Import binding done");

   /* ---------------------- */
   /* set authorization info */
   /* ---------------------- */

   rpc_binding_set_auth_info(
      bh,
      SERVER_NAME,
      rpc_c_protect_level_pkt_integ,    /* send checksum       */
      rpc_c_authn_default,
      NULL,                             /* use login context   */
      rpc_c_authz_dce,                  /* send PAC            */
      &status
   );

   check_DCE_status(status,
      "Set authorisation info");

}

/*********************************************************************/
/*                                                                   */
/* Name:        check_DCE_status                                     */
/*                                                                   */
/* Description: check_DCE_status() implements status checking routine.*/
/*                                                                   */
/*********************************************************************/

void  check_DCE_status(
   error_status_t   status,
   char             *msg)
{
   if ( status != error_status_ok )
   {
      printf("%s - ERROR\n", msg);

      dce_error_inq_text(
         status,
         dce_error_string,
         &error_inq_st);

      printf( "%s\n", dce_error_string);
      fflush(NULL);

      exit ( status );
   }
}
```

Figure 22 (Part 5 of 5). C Routine, PTSTC

B.3.6 PTSTS

```
/*********************************************************************/
/*                                                                   */
/* MODULE NAME: PTSTS                                                */
/*                                                                   */
/* DESCRIPTIVE NAME: Test and demonstration of  RACF passticket.     */
/*                                                                   */
/* FUNCTION:                                                         */
/*                                                                   */
/*     This module is a server module that will test the use of      */
/*     RACF passticket for a client.  This will test that the        */
/*     client has access to data sets that the server cannot         */
/*     access.                                                       */
/*                                                                   */
/*     The manager code, "ptst", adds a record to the nominated      */
/*     MVS data set.                                                 */
/*                                                                   */
/*     The server advertises its presence in the CDS by an entry     */
/*     of the <sysid> in the '/.:/passticket/ptst/' directory.       */
/*                                                                   */
/*     For example:                                                  */
/*                                                                   */
/*             '/.:/passticket/ptst/SY1'                             */
/*                                                                   */
/*     where the <sysid> is system name in the MVS CVT field,        */
/*     CVTSNAME. This is set by the installation in 'SYS1.PARMLIB'   */
/*     IEASYS00 member, SYSNAME parameter.                           */
/*                                                                   */
/*  PROCESSOR: C                                                     */
/*                                                                   */
/*  ENTRY POINT:                                                     */
/*                                                                   */
/*         PTSTS                                                     */
/*                                                                   */
/*  ENTERED FROM:                                                    */
/*                                                                   */
/*         MVS initiator, TSO CALL command, OpenEdition Shell        */
/*                                                                   */
/*  PARAMETERS:                                                      */
/*                                                                   */
/*         none                                                      */
/*                                                                   */
/*  EXTERNAL REFERENCES:                                             */
/*             ROUTINES:                                             */
/*                                                                   */
/*  EXITS: NORMAL--                                                  */
/*         ERROR---                                                  */
/*                                                                   */
/*  MACROS/INCLUDES USED:                                            */
/*                                                                   */
/*  ATTRIBUTES:                                                      */
/*                                                                   */
/*         none                                                      */
/*                                                                   */
/*  NOTES:                                                           */
/*                                                                   */
/*  1. This server issues a DCE login for the DCE principal name     */
/*     "ptst".                                                       */
/*                                                                   */
/*  2. A keytab file must be created for "ptst".                     */
/*                                                                   */
/*  3. "ptst" must be a member of:                                   */
/*         "subsys/dce/rpc-server-group".                            */
/*                                                                   */
/*  4. "ptst" must have:                                             */
/*                                                                   */
```

Figure 23 (Part 1 of 12). C Routine, PTSTS

```
/*        "rwdtci" authority for "/.:/passticket/ptst"              */
/*        "rwdtc"  authority for "/.:/passticket/ptst/SY1"          */
/*                                                                   */
/*  5. This code uses the C tri graph characters for:              */
/*                                                                   */
/*         left  square bracket   ??(                               */
/*         right square bracket   ??)                               */
/*                                                                   */
/*     to avoid possible character set complications on 3270        */
/*     terminals/emulators.                                         */
/*                                                                   */
/*                                                                   */
/*  CHANGE ACTIVITY:                                                */
/*                                                                   */
/*  20/2/95 Module created                        Eric Finkelstein  */
/*                                                                   */
/*******************************************************************/

/* ---------------------- */
/* pragmas                */
/* ---------------------- */

#pragma linkage(DCERACF,OS)
#pragma runopts(posix(on))

/* -------------------------------------- */
/*  included headers                      */
/* -------------------------------------- */

#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/time.h>
#include <pthread.h>
#include <exc@hand.h>
#include <dce/rpc.h>
#include <dce/sec_login.h>
#include <dce/keymgmt.h>
#include <dce/binding.h>
#include <dce/acct.h>
#include "dceracf.h"
#include "ptst.h"

/* -------------------------------------- */
/*  defined constants                     */
/* -------------------------------------- */

#define MAX_CONC_CALLS_PROTSEQ      2
#define MAX_CONC_CALLS_TOTAL        4
#define IF_HANDLE                   PTST_v1_0_s_ifspec
#define CDS_DIRECTORY               "/.:/passticket/ptst/"
#define SERVER_PRINCIPAL_NAME       "ptst"
#define DEBUG 0

/* -------------------------------------- */
/*  prototypes                            */
/* -------------------------------------- */

void  check_DCE_status(error_status_t status, char *msg);
void  refresh_login_context_rtn();

/* -------------------- */
/*  Global variables    */
/* -------------------- */

char    keytab[128];

/* -------------------------------------- */
/*  Main                                  */
/* -------------------------------------- */
```

Figure 23 (Part 2 of 12). C Routine, PTSTS

```
int main (int argc, char *argv[])
{

   /* -------------------------------- */
   /* local variables                  */
   /* -------------------------------- */

   unsigned_char_t       server_entry_name[129];
   unsigned_char_t       uuid_string[129];
   rpc_binding_vector_t  *bind_vector_p;
   sec_login_handle_t    login_context;
   sec_login_auth_src_t  auth_src;
   pthread_t             refresh_login_context_thread;
   void                  *server_key;
   boolean32             identity_valid;
   boolean32             reset_passwd;
   error_status_t        status;
   char                  sysid[9];
   char                  msg[128];
   int                   rc;

   /* -------------------------------- */
   /* setup identity in a login context */
   /* -------------------------------- */

   sec_login_setup_identity(
      SERVER_PRINCIPAL_NAME,
      sec_login_no_flags,
      &login_context,
      &status);

   check_DCE_status(status,
      "Setting Identity");

   /* ------------------------ */
   /* Get Key From KEYTAB File */
   /* ------------------------ */

   if (argc != 2)
   {
      printf("Keytab file not entered as a parameter\n");
      exit(1);
   }

   strcpy(keytab, argv[1]);
   printf("Keytab=%s\n", keytab);

   sec_key_mgmt_get_key(
      rpc_c_authn_dce_secret,
      keytab,
      SERVER_PRINCIPAL_NAME,
      0,
      &server_key,
      &status);

   check_DCE_status(status,
      "Getting Key from Keytab file");

   /* ---------------------- */
   /* Validate the Identity  */
   /* ---------------------- */

   identity_valid =
      sec_login_validate_identity(
         login_context,
         server_key,
         &reset_passwd,
         &auth_src,
         &status);
```

Figure 23 (Part 3 of 12). C Routine, PTSTS

```
check_DCE_status(status,
   "Validate Identity");

/* ----------------- */
/* Certify identity  */
/* ----------------- */

sec_login_certify_identity(
   login_context,
   &status);

check_DCE_status(status,
   "Certifying identity");

/* ----------------- */
/* Set login context */
/* ----------------- */

sec_login_set_context(
   login_context,
   &status);

check_DCE_status(status,
   "Setting context");

/* ---------------- */
/* Free Key Storage */
/* ---------------- */

sec_key_mgmt_free_key(
   &server_key,
   &status);

check_DCE_status(status,
   "Freeing key storage");

/* ------------------------------- */
/* check login OK and authenicated */
/* ------------------------------- */

if (identity_valid &&
   (auth_src == sec_login_auth_src_network))
{
  printf("Logged in as principal=%s\n", SERVER_PRINCIPAL_NAME);
  fflush (NULL);
}
else
{
  printf("Failed to login  as principal=%s\n",
     SERVER_PRINCIPAL_NAME);
  fflush (NULL);
  exit(1);
}

/* ---------------------------------------------- */
/* set the authentication level which will be used */
/* ---------------------------------------------- */

rpc_server_register_auth_info (
   SERVER_PRINCIPAL_NAME,
   rpc_c_authn_dce_secret,
   NULL,
   keytab,
   &status
);

check_DCE_status( status,
   "Setting authentication level");
```

Figure 23 (Part 4 of 12). C Routine, PTSTS

```
/* ------------------------------------------------------ */
/* Register interface/epv associations with rpc runtime. */
/* ------------------------------------------------------ */

rpc_server_register_if (
   IF_HANDLE,
   NULL,
   NULL,
   &status );

check_DCE_status( status,
   "Registering interface/epv");

/* ------------------------------------------------------------ */
/* Inform rpc runtime to use all supported protocol sequences. */
/* ------------------------------------------------------------ */

rpc_server_use_all_protseqs (
   MAX_CONC_CALLS_PROTSEQ,
   &status );

check_DCE_status( status,
   "Setting protocol sequences");

/* ------------------------------------------------ */
/* Ask the runtime which binding handle will be used. */
/* ------------------------------------------------ */


rpc_server_inq_bindings(
   &bind_vector_p,
   &status );

check_DCE_status( status,
   "Getting binding handle");

/* ---------------------------------------------- */
/* Register binding information with endpoint map */
/* ---------------------------------------------- */

rpc_ep_register(
   IF_HANDLE,
   bind_vector_p,
   NULL,
   "PTST server, version 1.0",
   &status);

check_DCE_status( status,
   "Registering endpoint");

/* -------------------- */
/* Get <sysid> from CVT */
/* -------------------- */

rc = DCERACF(RACF_get_MVS_sysid, sysid);

/* ---------------------- */
/* build server entry name */
/* ---------------------- */

strcpy(server_entry_name, CDS_DIRECTORY);
strcat(server_entry_name, sysid);

/* ------------------------------------ */
/* Export binding info to the namespace. */
/* ------------------------------------ */

rpc_ns_binding_export (
   rpc_c_ns_syntax_default,
   server_entry_name,
```

Figure 23 (Part 5 of 12). C Routine, PTSTS

```
   IF_HANDLE,
   bind_vector_p,
   NULL,
   &status
);

check_DCE_status( status,
   "Exporting binding information");

/* --------------------------- */
/* Start refresh login thread   */
/* --------------------------- */

rc = pthread_create(
   &refresh_login_context_thread,
   pthread_attr_default,
   (pthread_startroutine_t) refresh_login_context_rtn,
   (pthread_addr_t) login_context);

if (rc == -1)
{
   printf("Failed to create refresh login context thread\n");
   exit(1);
}

/* --------------------------- */
/* Listen for service requests. */
/* --------------------------- */

TRY {

   printf("Server entry name=%s\n",
           server_entry_name);

   printf( "Server listening\n");
   fflush(NULL);

   rpc_server_listen (
      MAX_CONC_CALLS_TOTAL,
      &status );

   check_DCE_status( status,
      "Listening for service requests");

}

/* ---------------------------------------- */
/* shutdown request from an authorised client */
/* ---------------------------------------- */

FINALLY {

   /* ------------------------------------------------- */
   /* Unexport the binding information from the namespace. */
   /* ------------------------------------------------- */

   rpc_ns_binding_unexport (
      rpc_c_ns_syntax_default,
      server_entry_name,
      IF_HANDLE,
      NULL,
      &status);

   check_DCE_status( status,
      "Unexporting binding information");

   /* ---------------------------------- */
   /* Unregister interface from RPC runtime */
   /* ---------------------------------- */
```

Figure 23 (Part 6 of 12). C Routine, PTSTS

```
      rpc_server_unregister_if (
         IF_HANDLE,
         NULL,
         &status );

      check_DCE_status( status,
         "Unregistering interface");

      /* ---------------------------- */
      /* Unregister interface from EPV */
      /* ---------------------------- */

      rpc_ep_unregister (
         IF_HANDLE,
         bind_vector_p,
         NULL,
         &status );

      check_DCE_status( status,
         "Unregistering interface from EPV");

      /* ------------------------------ */
      /* exit                           */
      /* ------------------------------ */

      exit ( 0 );
   }
   ENDTRY;
}

/*************************************************************************/
/*                                                                       */
/* Function:    refresh_login_context_rtn                                */
/*                                                                       */
/* Description: Every hour, refresh login context                        */
/*                                                                       */
/*************************************************************************/

void refresh_login_context_rtn(
        sec_login_handle_t  login_context)
{

   /* ------------------------------ */
   /* local variables                */
   /* ------------------------------ */

   signed32         current_time;
   signed32         expiration;
   signed32         delay_time;
   struct timespec  delay;
   unsigned32       used_kvno;
   boolean32        reset_password;
   boolean32        identity_valid;
   void             *server_key;
   sec_login_auth_src_t  auth_src;
   error_status_t   status;
   #define          MINUTE 60
   #define          DAYSECS 86400

   /* ------------------- */
   /* infinite loop       */
   /* ------------------- */

   while (1)
   {

      /* ---------------------------------------- */
      /* wait till logon context is about to expire */
      /* ---------------------------------------- */
```

Figure 23 (Part 7 of 12). C Routine, PTSTS

```
sec_login_get_expiration (
   login_context,
   &expiration,
   &status);

if ((status != rpc_s_ok) &&
    (status != sec_login_s_not_certified))
{
   printf("Cannot get login context expiration time\n");
   fflush (NULL);
   exit(1);
}

current_time = time(NULL);
delay_time = expiration - current_time - (10*MINUTE);

if (delay_time > 0)
{
   delay.tv_sec = delay_time;
   delay.tv_nsec = 0;
   pthread_delay_np(&delay);
}

/* ---------------------------- */
/* refresh identity             */
/* ---------------------------- */

sec_login_refresh_identity(
   login_context,
   &status);

check_DCE_status(status,
   "Refreshing identity file");

/* ------------------------ */
/* Get Key From KEYTAB File */
/* ------------------------ */

sec_key_mgmt_get_key(
   rpc_c_authn_dce_secret,
   keytab,
   SERVER_PRINCIPAL_NAME,
   0,
   &server_key,
   &status);

check_DCE_status(status,
   "Getting Key from Keytab file");

/* ---------------------- */
/* Validate the Identity  */
/* ---------------------- */

identity_valid =
   sec_login_validate_identity(
      login_context,
      server_key,
      &reset_password,
      &auth_src,
      &status);

check_DCE_status(status,
   "Validating Identity");

/* ---------------- */
/* Free Key Storage */
/* ---------------- */

sec_key_mgmt_free_key(
   &server_key,
```

Figure 23 (Part 8 of 12). C Routine, PTSTS

```
         &status);

      check_DCE_status(status,
         "Freeing key storage");

      /* ---------------- */
      /* check refresh OK */
      /* ---------------- */

      if (!identity_valid)
      {
        printf("Refresh of login context failed\n");
        fflush (NULL);
      }
   }
}

/*********************************************************************/
/*                                                                   */
/* Name:        check_DCE_status                                     */
/*                                                                   */
/* Description: check_DCE_status() implements status checking routine.*/
/*                                                                   */
/*********************************************************************/

void  check_DCE_status(error_status_t status, char *msg)
{

   /* ------------------- */
   /* local variables     */
   /* ------------------- */

   char            dce_error_string[256];
   error_status_t  error_inq_st;

   /* ------------------- */
   /* check status        */
   /* ------------------- */

   if ( status == error_status_ok )
   {
      if (DEBUG == 1)
      {
         printf("%s - OK\n", msg);
      }
   }
   else
   {
      printf("%s - ERROR\n", msg);

      dce_error_inq_text(
         status,
         dce_error_string,
         &error_inq_st);

      printf( "%s\n", dce_error_string);
      fflush(NULL);

      exit ( status );
   }
}

/*********************************************************************/
/*                                                                   */
/* Function:    ptst                                                 */
/*                                                                   */
/* Description: add a record to MVS data set.                        */
/*                                                                   */
/*********************************************************************/
```

Figure 23 (Part 9 of 12). C Routine, PTSTS

```
idl_long_int ptst(
          handle_t    bh,
          char        *ticket,
          char        *dsn,
          char        *record)
{

   /* ------------------------------ */
   /* local variables                */
   /* ------------------------------ */

   idl_long_int     logoff_rc;
   idl_long_int     rc;
   char             msg[128];
   char             class[9];
   char             entity[64];
   char             access[10];
   char             principal[128];
   error_status_t   status;
   FILE             *fh;
   char             filename[50];

   /* ------------------- */
   /* message definitions */
   /* ------------------- */

   char *ptst_msg[] = {
      /*0*/ "Record added succesfully",
      /*1*/ "Record not added, password or RACF userid invalid",
      /*2*/ "PTST server stopped",
      /*3*/ "You are not authorized to stop the PTST server",
      /*4*/ "Unable to open file",
      /*x*/ ""        };

   /* -------------------------- */
   /* logon client onto RACF     */
   /* -------------------------- */

   rc = DCERACF(RACF_logon, bh, ticket);

   if (rc != 0)
   {
      RACF_logon_error(rc, msg);
      printf("%s - rc=%d\n", msg, rc);
      fflush(NULL);
      rc = 1;
   }
   else
   {

      /* -------------------- */
      /* get DCE principal    */
      /* -------------------- */

      rc = DCERACF(RACF_get_DCE_principal, principal);

      /* -------------------------- */
      /* is this a stop request  */
      /* -------------------------- */

      if (strcmp(dsn, "STOP") == 0)
      {

         /* ------------------------------------------- */
         /* is the client authorised to stop the server? */
         /* ------------------------------------------- */

         strcpy(class,"$dceracf");
         strcpy(entity,"administrator");
         strcpy(access,"read");
```

Figure 23 (Part 10 of 12). C Routine, PTSTS

```
      rc = DCERACF(RACF_check_authorisation,
                   bh,
                   class,
                   entity,
                   access);

      if (rc == 0)
      {
         printf("PTST server has been requested to stop by %s\n",
                principal);
         rpc_mgmt_stop_server_listening(NULL, &status);
         fflush(NULL);
         rc = 2;
      }
      else
      {
         RACF_check_authorisation_error(rc, msg);
         printf("SAVEPW server has been requested to stop by %s, ",
                principal);
         printf("%s - rc=%d\n", msg, rc);
         fflush(NULL);
         rc = 3;
      }
   }
   else
   {

      /* ----------------- */
      /* Open the data set */
      /* ----------------- */

      strcpy(filename, "//'");
      strcat(filename, dsn);
      strcat(filename, "'");
      printf("filename=%s\n",filename);
      fflush(NULL);

      fh = fopen(filename,
           "ab, recfm=VB, lrecl=255, blksize=2554, type=record");
      if (fh == NULL)
      {
         printf("Unable to open file %s\n", filename);
         rc = 4;
      }
      else
      {
         /* ----------------- */
         /* write record      */
         /* ----------------- */

         fwrite(record, strlen(record), 1, fh);

         /* ----------------- */
         /* close file        */
         /* ----------------- */

         fclose(fh);

      }
   }

   /* ------------------------- */
   /* logoff RACF               */
   /* ------------------------- */

   logoff_rc = DCERACF(RACF_logoff, bh);

   if (logoff_rc != 0)
   {
```

Figure 23 (Part 11 of 12). C Routine, PTSTS

```
         RACF_logoff_error(logoff_rc, msg);
         printf("%s - rc=%d\n", msg, logoff_rc);
         fflush(NULL);
      }
   }

   return(rc);

}
```

Figure 23 (Part 12 of 12). C Routine, PTSTS

B.3.7 SAVEPW

```
/*********************************************************************/
/*                                                                   */
/* MODULE NAME: SAVEPW                                               */
/*                                                                   */
/* DESCRIPTIVE NAME: Change DCE password in RACF and DCE registry.   */
/*                                                                   */
/* FUNCTION:                                                         */
/*                                                                   */
/*     This module is a client module that will allow                */
/*     a DCE principal to change his/her DCE password saved in       */
/*     his/her associated RACF user profile. To ensure that the      */
/*     password is same in the DCE registry, the password in also    */
/*     updated in the DCE registry.                                  */
/*                                                                   */
/*     There may be several RACF hosts associated with the DCE       */
/*     principal. The SVPWS server will determine the RACF userid    */
/*     associated with this DCE principal from the group list        */
/*     sent in the DCE PAC (Privilege Attribute Certificate). It will*/
/*     scan the PAC for a group of RACF_<sysid>_<userid> and extract */
/*     the RACF userid from that name. The DCE principal must be     */
/*     the one and only member of the above DCE group.               */
/*                                                                   */
/*  PROCESSOR: C                                                     */
/*                                                                   */
/*  ENTRY POINT:                                                     */
/*                                                                   */
/*         SAVEPW                                                    */
/*                                                                   */
/*  ENTERED FROM:                                                    */
/*                                                                   */
/*         MVS initiator, TSO CALL command, OpenEdition Shell        */
/*                                                                   */
/*  SYNTAX:                                                          */
/*                                                                   */
/*         SAVEPW sysid old_password new_password verify_password    */
/*                                                                   */
/*  EXTERNAL REFERENCES:                                             */
/*             ROUTINES:                                             */
/*                                                                   */
/*  EXITS: NORMAL--                                                  */
/*         ERROR---                                                  */
/*                                                                   */
/*  MACROS/INCLUDES USED:                                            */
/*                                                                   */
/*  ATTRIBUTES:                                                      */
/*                                                                   */
/*  NOTES:                                                           */
/*                                                                   */
/*  CHANGE ACTIVITY:                                                 */
/*                                                                   */
/*  15/1/95 Module created                          Eric Finkelstein  */
/*                                                                   */
/*********************************************************************/

/* ------------------------ */
/* pragmas                  */
/* ------------------------ */

#pragma runopts(posix(on))
#pragma linkage(DCERACF,OS)

/* ------------------------ */
/* included headers         */
/* ------------------------ */
```

Figure 24 (Part 1 of 8). C Routine, SAVEPW

```
#include <string.h>
#include <stdio.h>
#include <dce/acct.h>
#include <dce/binding.h>
#include <dce/sec_login.h>
#include <dce/pgo.h>
#include <dce/uuid.h>
#include "dceracf.h"
#include "svpw.h"

/* ------------------------ */
/* defined constants        */
/* ------------------------ */

#define IF_HANDLE        SVPW_v1_0_c_ifspec
#define CDS_DIRECTORY    "/.:/dceracf/"
#define SERVER_NAME      "savepw"
#define DEBUG            0

/* ---------------------- */
/*  message definitions  */
/* ---------------------- */

#define OK                              0
#define SYNTAX_ERROR                    1
#define DIFFERENT_PASSWORDS_ERROR       2
#define SERVER_ERROR                    3
#define STOP                            0

/* ------------------------ */
/* prototypes               */
/* ------------------------ */

void message(int msg);
void help ();
void change_DCE_password();
void check_DCE_status(error_status_t status, char *msg);

/* ------------------------ */
/* global variables         */
/* ------------------------ */

int               i;
int               rc;
handle_t          bh;
char              msg[128];
char              server_entry_name[128];
char              sysid[9];
char              principal[129];
char              password[129];
char              old_password[129];
char              new_password[129];
char              verify_password[129];
char              dce_error_string[256];
char              registry_site[256];
char              *uuid_string;
sec_login_handle_t          *login_context;
sec_login_net_info_t        net_info;
rpc_ns_handle_t             ns_handle;
error_status_t              status;
error_status_t              error_inq_st;
sec_rgy_handle_t            registry_handle;
sec_passwd_rec_t            new_pwrec;
sec_passwd_rec_t            old_pwrec;
sec_passwd_version_t        new_password_version;
sec_rgy_login_name_t        login_name;

/* ------------------------ */
/* SAVEPW                   */
/* ------------------------ */
```

Figure 24 (Part 2 of 8). C Routine, SAVEPW

```
int main (int argc, char *argv[])
{

   /* ---------------------- */
   /* parse the command line */
   /* ---------------------- */

   if (argc == 5)
   {

      strcpy(sysid, argv[1]);

      for (i=0; i <= strlen(sysid); i++)
      {
         sysid[i] = toupper(sysid[i]);
      }

      strcpy(old_password, argv[2]);
      strcpy(new_password, argv[3]);
      strcpy(verify_password, argv[4]);

   }
   else
   {
      if ((argc == 3) &&
          (strcmp(argv[2], "stop") == 0))
      {
         strcpy(sysid, argv[1]);

         for (i=0; i <= strlen(sysid); i++)
         {
            sysid[i] = toupper(sysid[i]);
         }

         strcpy(old_password, "-s-t-o-p-");
         strcpy(new_password, "-s-t-o-p-");
         strcpy(verify_password, "-s-t-o-p-");
      }
      else
      {
         message(SYNTAX_ERROR);
         exit(SYNTAX_ERROR);
      }
   }

   /* ---------------------- */
   /* check passwords        */
   /* ---------------------- */

   if (strcmp(new_password, verify_password) != 0)
   {
      message(DIFFERENT_PASSWORDS_ERROR);
      exit(DIFFERENT_PASSWORDS_ERROR);
   }

   /* --------------------------- */
   /* build cds server entry name  */
   /* --------------------------- */

   strcpy(server_entry_name, CDS_DIRECTORY);
   strcat(server_entry_name, sysid);

   /* ------------------------------------ */
   /* set up for reading binding information */
   /* ------------------------------------ */

   rpc_ns_binding_import_begin(
      rpc_c_ns_syntax_default,
      server_entry_name,
```

Figure 24 (Part 3 of 8). C Routine, SAVEPW

```
   IF_HANDLE,
   NULL,
   &ns_handle,
   &status
);

check_DCE_status(status,
   "Binding import begin");

/* ------------------------------------------------ */
/* import first binding information from name space */
/* ------------------------------------------------ */

rpc_ns_binding_import_next(
   ns_handle,
   &bh,
   &status
);

check_DCE_status(status,
   "Import binding");

/* -------------------------------- */
/* stop reading binding indormation */
/* -------------------------------- */

rpc_ns_binding_import_done(
   &ns_handle,
   &status
);

check_DCE_status(status,
   "Import binding done");

/* ---------------------- */
/* set authorization info */
/* ---------------------- */

rpc_binding_set_auth_info(
   bh,
   SERVER_NAME,
   rpc_c_protect_level_pkt_integ,    /* send checksum       */
   rpc_c_authn_default,
   NULL,                             /* use login context   */
   rpc_c_authz_dce,                  /* send PAC            */
   &status
);


check_DCE_status(status,
   "Set authorisation info");

/* ------------------------------------ */
/* call the remote procedure            */
/* ------------------------------------ */

rc = svpw(bh, new_password);

if (rc != 0)                        /* if error on server */
{
   i = rc >> 16;                    /* extract  routine number */
   rc = rc << 16;                   /* remove routine number   */
   rc = rc >> 16;                   /* put rc back to bottom   */

   switch ( i ) {
   case RACF_logon:
      RACF_logon_error(rc, msg);
      break;

   case RACF_check_authorisation:
```

Figure 24 (Part 4 of 8). C Routine, SAVEPW

```
         RACF_check_authorisation_error(rc, msg);
         printf("%s rc=%d\n", msg, rc);
         printf("You are not authorised to stop the SAVEPW server\n");
         printf("SAVEPW server has not been stopped\n");
         fflush(NULL);
         exit(1);
         break;

      case RACF_set_DCE_password:
         RACF_set_DCE_password_error(rc, msg);
         break;

      case RACF_logoff:
         RACF_logoff_error(rc, msg);
         break;

      case STOP:
         printf("SAVEPW server stopped\n");
         fflush(NULL);
         exit(1);

      default:
         printf("something is wrong\n");

      }
      printf("%s rc=%d\n", msg, rc);

      message(SERVER_ERROR);
      exit(SERVER_ERROR);
   }

   /* ------------------------------------ */
   /* force same password on DCE           */
   /* ------------------------------------ */

   change_DCE_password();

   printf("DCE password changed in both RACF and DCE\n");

   exit ();
}

/*********************************************************************/
/*                                                                   */
/* Name:        check_DCE_status                                     */
/*                                                                   */
/* Description: check_DCE_status() implements status checking routine.*/
/*                                                                   */
/*********************************************************************/

void  check_DCE_status(
   error_status_t   status,
   char             *msg)
{
   if ( status == error_status_ok )
   {
      if (DEBUG == 1)
      {
         printf("%s - OK\n", msg);
      }
   }
   else
   {
      printf("%s - ERROR\n", msg);

      dce_error_inq_text(
         status,
         dce_error_string,
         &error_inq_st);
```

Figure 24 (Part 5 of 8). C Routine, SAVEPW

```
      printf( "%s\n", dce_error_string);
      fflush(NULL);

      exit ( status );
   }
}

/*********************************************************************/
/*                                                                   */
/* Name:        change_DCE_password()                                */
/*                                                                   */
/* Description: Implements the DCE password change                   */
/*                                                                   */
/*********************************************************************/

void  change_DCE_password()
{

   /* ---------------------------- */
   /* open an update registry site  */
   /* ---------------------------- */

   sec_rgy_site_open_update(
      registry_site,                      /* find local registry */
      &registry_handle,                   /* registry context    */
      &status);                           /* return code         */

   check_DCE_status(status,
      "Binding to registry");

   /* ---------------------------- */
   /* get current logon context     */
   /* ---------------------------- */

   sec_login_get_current_context(
      &login_context,
      &status );

   check_DCE_status(status,
      "Getting current login context");

   /* ---------------------------- */
   /* certify the context           */
   /* ---------------------------- */

   sec_login_certify_identity(
      login_context,
      &status);

   check_DCE_status(status,
      "Certifying login context");

   /* ---------------------------- */
   /* inquire net info              */
   /* ---------------------------- */

   sec_login_inquire_net_info(
      login_context,
      &net_info,
      &status );

   check_DCE_status(status,
      "Inquire net info");

   /* ---------------------------- */
   /* get principal name using UUID */
   /* ---------------------------- */

   sec_rgy_pgo_id_to_name(
      registry_handle,
```

Figure 24 (Part 6 of 8). C Routine, SAVEPW

```
      sec_rgy_domain_person,
      &net_info.pac.principal.uuid,
      &login_name,
      &status );

   check_DCE_status(status,
      "Principal name from UUID");

   /* -------------------------------- */
   /* create caller key password record  */
   /* -------------------------------- */

   old_pwrec.key.tagged_union.plain = old_password;
   old_pwrec.key.key_type = sec_passwd_plain;
   old_pwrec.pepper = NULL;
   old_pwrec.version_number = sec_passwd_c_version_none;

   /* ---------------------------- */
   /* create new password record    */
   /* ---------------------------- */

   new_pwrec.key.tagged_union.plain = new_password;
   new_pwrec.key.key_type = sec_passwd_plain;
   new_pwrec.pepper = NULL;
   new_pwrec.version_number = sec_passwd_c_version_none;

   /* ---------------------------- */
   /* change the password           */
   /* ---------------------------- */

   sec_rgy_acct_passwd(
      registry_handle,
      &login_name,
      &old_pwrec,
      &new_pwrec,
      sec_passwd_des,
      &new_password_version,
      &status);

   check_DCE_status(status,
      "Changing password in the registry");

   /* ---------------------------- */
   /* free net_info                 */
   /* ---------------------------- */

   sec_login_free_net_info(
      net_info );

   check_DCE_status(status,
      "Freeing net info");

   /* ---------------------------- */
   /* unbind registry site          */
   /* ---------------------------- */

   sec_rgy_site_close(
      registry_handle,
      &status);

   check_DCE_status(status,
      "Closing registry");
}

/*********************************************************************/
/*                                                                   */
/* Name:        message                                              */
/*                                                                   */
/* Description: message implements the error printing routine.       */
/*                                                                   */
```

Figure 24 (Part 7 of 8). C Routine, SAVEPW

```
/*********************************************************************/

void message(int msg)
{
   switch ( msg ) {
   case SYNTAX_ERROR:
      printf("SAVEPW - Incorrect number of parameters\n");
      help();
      break;

   case DIFFERENT_PASSWORDS_ERROR:
      printf("SAVEPW - password and verify password do not match\n");
      help();
      break;

   case SERVER_ERROR:
      printf("SAVEPW - password was not updated in RACF\n");
      break;

   default:
      printf("SAVEPW - something is wrong\n");

   }

}

/*********************************************************************/
/*                                                                   */
/* Name:        help()                                               */
/*                                                                   */
/* Description: help() implements the error printing routine.        */
/*                                                                   */
/*********************************************************************/

void help ()
{
   printf(" \nSAVEPW Syntax:\n");
   printf(" \n");
   printf(" \nSAVEPW <sysid> <old password> <new_password> <verify password>");
   printf(" \n");
}
```

Figure 24 (Part 8 of 8). C Routine, SAVEPW

B.3.8 SAVEPWS

```
/*********************************************************************/
/*                                                                   */
/* MODULE NAME: SAVEPWS                                              */
/*                                                                   */
/* DESCRIPTIVE NAME: A DCE server to change DCE password in RACF     */
/*                   user profile.                                   */
/*                                                                   */
/* FUNCTION:                                                         */
/*                                                                   */
/*     This module is a server module that will allow                */
/*     a DCE principal to change his/her DCE password saved in       */
/*     his/her associated RACF user profile.                         */
/*                                                                   */
/*     There may be several RACF hosts associated with the DCE       */
/*     principal. This server will determine the RACF userid         */
/*     associated with this DCE principal from the group list        */
/*     sent in the DCE PAC (Privilege Attributes Certificate).       */
/*     It will scan the PAC for a group of RACF_<sysid>_<userid>     */
/*     and extract the RACF userid from that name. The DCE           */
/*     principal must be the one and only member of the above DCE    */
/*     group.                                                        */
/*                                                                   */
/*     The server advertises its presence in the CDS by an entry     */
/*     of the <sysid> in the '/.:/dceracf/' directory. For example   */
/*                                                                   */
/*             '/.:/dceracf/SY1'                                     */
/*                                                                   */
/*     where the <sysid> is system name in the MVS CVT field,        */
/*     CVTSNAME. This is set by the installation in 'SYS1.PARMLIB'   */
/*     IEASYS00 member, SYSNAME parameter.                           */
/*                                                                   */
/*  PROCESSOR: C                                                     */
/*                                                                   */
/*  ENTRY POINT:                                                     */
/*                                                                   */
/*         SAVEPWS                                                   */
/*                                                                   */
/*  ENTERED FROM:                                                    */
/*                                                                   */
/*         MVS initiator, TSO CALL command, OpenEdition Shell        */
/*                                                                   */
/*  PARAMETERS:                                                      */
/*                                                                   */
/*         none                                                      */
/*                                                                   */
/*  EXTERNAL REFERENCES:                                             */
/*             ROUTINES:                                             */
/*                                                                   */
/*  EXITS: NORMAL--                                                  */
/*         ERROR---                                                  */
/*                                                                   */
/*  MACROS/INCLUDES USED:                                            */
/*                                                                   */
/*  ATTRIBUTES:                                                      */
/*                                                                   */
/*  NOTES:                                                           */
/*                                                                   */
/*  1. This server issues a DCE login for the DCE principal name     */
/*     "savepw".                                                     */
/*                                                                   */
/*  2. A keytab file must be created for "savepw".                   */
/*                                                                   */
/*  3. "savepw" must be a member of "subsys/dce/rpc-server-group".   */
/*                                                                   */
/*  4. "savepw" must have:                                           */
```

Figure 25 (Part 1 of 12). C Routine, SAVEPWS

```
/*                                                                    */
/*       "rwdtci" authority for "/.:/dceracf"                         */
/*       "rwdtc"  authority for "/.:/dceracf/<sysid>"                 */
/*                                                                    */
/*                                                                    */
/*  CHANGE ACTIVITY:                                                  */
/*                                                                    */
/*  15/1/95 Module created                        Eric Finkelstein    */
/*                                                                    */
/**********************************************************************/

/* ---------------------- */
/* pragmas                */
/* ---------------------- */

#pragma linkage(DCERACF,OS)
#pragma runopts(posix(on))

/* -------------------------------------- */
/*  included headers                      */
/* -------------------------------------- */

#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/time.h>
#include <pthread.h>
#include <exc@hand.h>
#include <dce/rpc.h>
#include <dce/sec_login.h>
#include <dce/keymgmt.h>
#include <dce/binding.h>
#include <dce/acct.h>
#include "dceracf.h"
#include "svpw.h"

/* -------------------------------------- */
/*  defined constants                     */
/* -------------------------------------- */

#define MAX_CONC_CALLS_PROTSEQ      2
#define MAX_CONC_CALLS_TOTAL        4
#define IF_HANDLE                   SVPW_v1_0_s_ifspec
#define CDS_DIRECTORY               "/.:/dceracf/"
#define SERVER_PRINCIPAL_NAME       "savepw"
#define DEBUG 0

/* -------------------------------------- */
/*  prototypes                            */
/* -------------------------------------- */

void  check_DCE_status(error_status_t status, char *msg);
void  refresh_login_context_rtn();

/* -------------------- */
/*  Global variables    */
/* -------------------- */

char    keytab[128];

/* -------------------------------------- */
/*  Main                                  */
/* -------------------------------------- */

int main (int argc, char *argv[])
{

   /* ------------------------------- */
   /* local variables                 */
   /* ------------------------------- */
```

Figure 25 (Part 2 of 12). C Routine, SAVEPWS

```
unsigned_char_t       server_entry_name[129];
unsigned_char_t       uuid_string[129];
rpc_binding_vector_t  *bind_vector_p;
sec_login_handle_t    login_context;
sec_login_auth_src_t  auth_src;
pthread_t             refresh_login_context_thread;
void                  *server_key;
boolean32             identity_valid;
boolean32             reset_passwd;
error_status_t        status;
char                  sysid[9];
char                  msg[128];
int                   rc;

/* -------------------------------- */
/* setup identity in a login context */
/* -------------------------------- */

sec_login_setup_identity(
   SERVER_PRINCIPAL_NAME,
   sec_login_no_flags,
   &login_context,
   &status);

check_DCE_status(status,
   "Setting Identity");

/* ------------------------ */
/* Get Key From KEYTAB File */
/* ------------------------ */

if (argc != 2)
{
   printf("Keytab file not entered as a parameter\n");
   exit(1);
}

strcpy(keytab, argv[1]);
printf("Keytab=%s\n", keytab);

sec_key_mgmt_get_key(
   rpc_c_authn_dce_secret,
   keytab,
   SERVER_PRINCIPAL_NAME,
   0,
   &server_key,
   &status);

check_DCE_status(status,
   "Getting Key from Keytab file");

/* ---------------------- */
/* Validate the Identity  */
/* ---------------------- */

identity_valid =
   sec_login_validate_identity(
      login_context,
      server_key,
      &reset_passwd,
      &auth_src,
      &status);

check_DCE_status(status,
   "Validate Identity");

/* ----------------- */
/* Certify identity  */
/* ----------------- */
```

Figure 25 (Part 3 of 12). C Routine, SAVEPWS

```
sec_login_certify_identity(
   login_context,
   &status);

check_DCE_status(status,
   "Certifying identity");

/* ----------------- */
/* Set login context */
/* ----------------- */

sec_login_set_context(
   login_context,
   &status);

check_DCE_status(status,
   "Setting context");

/* ---------------- */
/* Free Key Storage */
/* ---------------- */

sec_key_mgmt_free_key(
   &server_key,
   &status);

check_DCE_status(status,
   "Freeing key storage");

/* ------------------------------- */
/* check login OK and authenicated */
/* ------------------------------- */

if (identity_valid &&
   (auth_src == sec_login_auth_src_network))
{
  printf("Logged in as principal=%s\n", SERVER_PRINCIPAL_NAME);
  fflush (NULL);
}
else
{
  printf("Failed to login  as principal=%s\n",
     SERVER_PRINCIPAL_NAME);
  fflush (NULL);
  exit(1);
}

/* ---------------------------------------------- */
/* set the authentication level which will be used */
/* ---------------------------------------------- */

rpc_server_register_auth_info (
   SERVER_PRINCIPAL_NAME,
   rpc_c_authn_dce_secret,
   NULL,
   keytab,
   &status
);

check_DCE_status( status,
   "Setting authentication level");

/* ------------------------------------------------- */
/* Register interface/epv associations with rpc runtime. */
/* ------------------------------------------------- */

rpc_server_register_if (
   IF_HANDLE,
   NULL,
```

Figure 25 (Part 4 of 12). C Routine, SAVEPWS

```
   NULL,
   &status );

check_DCE_status( status,
   "Registering interface/epv");

/* ------------------------------------------------------------ */
/* Inform rpc runtime to use all supported protocol sequences. */
/* ------------------------------------------------------------ */

rpc_server_use_all_protseqs (
   MAX_CONC_CALLS_PROTSEQ,
   &status );

check_DCE_status( status,
   "Setting protocol sequences");

/* ------------------------------------------------- */
/* Ask the runtime which binding handle will be used. */
/* ------------------------------------------------- */


rpc_server_inq_bindings(
   &bind_vector_p,
   &status );

check_DCE_status( status,
   "Getting binding handle");

/* ---------------------------------------------- */
/* Register binding information with endpoint map */
/* ---------------------------------------------- */

rpc_ep_register(
   IF_HANDLE,
   bind_vector_p,
   NULL,
   "SAVEPW server, version 1.0",
   &status);

check_DCE_status( status,
   "Registering endpoint");

/* -------------------- */
/* Get <sysid> from CVT */
/* -------------------- */

rc = DCERACF(RACF_get_MVS_sysid, sysid);

/* ----------------------- */
/* build server entry name */
/* ----------------------- */

strcpy(server_entry_name, CDS_DIRECTORY);
strcat(server_entry_name, sysid);

/* ------------------------------------ */
/* Export binding info to the namespace. */
/* ------------------------------------ */

rpc_ns_binding_export (
   rpc_c_ns_syntax_default,
   server_entry_name,
   IF_HANDLE,
   bind_vector_p,
   NULL,
   &status
);

check_DCE_status( status,
```

Figure 25 (Part 5 of 12). C Routine, SAVEPWS

```
   "Exporting binding information");

/* -------------------------- */
/* Start refresh login thread   */
/* -------------------------- */

rc = pthread_create(
   &refresh_login_context_thread,
   pthread_attr_default,
   (pthread_startroutine_t) refresh_login_context_rtn,
   (pthread_addr_t) login_context);

if (rc == -1)
{
   printf("Failed to create refresh login context thread\n");
   exit(1);
}

/* -------------------------- */
/* Listen for service requests. */
/* -------------------------- */

TRY {

   printf("Server entry name=%s\n",
           server_entry_name);

   printf( "Server listening\n");
   fflush(NULL);

   rpc_server_listen (
      MAX_CONC_CALLS_TOTAL,
      &status );

   check_DCE_status( status,
      "Listening for service requests");

}

/* ---------------------------------------- */
/* shutdown request from an authorised client */
/* ---------------------------------------- */

FINALLY {

   /* ------------------------------------------------- */
   /* Unexport the binding information from the namespace. */
   /* ------------------------------------------------- */

   rpc_ns_binding_unexport (
      rpc_c_ns_syntax_default,
      server_entry_name,
      IF_HANDLE,
      NULL,
      &status);

   check_DCE_status( status,
      "Unexporting binding information");

   /* ---------------------------------- */
   /* Unregister interface from RPC runtime */
   /* ---------------------------------- */

   rpc_server_unregister_if (
      IF_HANDLE,
      NULL,
      &status );

   check_DCE_status( status,
      "Unregistering interface");
```

Figure 25 (Part 6 of 12). C Routine, SAVEPWS

```
      /* ---------------------------- */
      /* Unregister interface from EPV */
      /* ---------------------------- */

      rpc_ep_unregister (
         IF_HANDLE,
         bind_vector_p,
         NULL,
         &status );

      check_DCE_status( status,
         "Unregistering interface from EPV");

      /* ------------------------------- */
      /* exit                            */
      /* ------------------------------- */

      exit ( 0 );
   }
   ENDTRY;
}

/*************************************************************************/
/*                                                                       */
/* Function:    refresh_login_context_rtn                                */
/*                                                                       */
/* Description: Every hour, refresh login context                        */
/*                                                                       */
/*************************************************************************/

void refresh_login_context_rtn(
        sec_login_handle_t  login_context)
{

   /* ------------------------------ */
   /* local variables                */
   /* ------------------------------ */

   signed32         current_time;
   signed32         expiration;
   signed32         delay_time;
   struct timespec  delay;
   unsigned32       used_kvno;
   boolean32        reset_password;
   boolean32        identity_valid;
   void             *server_key;
   sec_login_auth_src_t  auth_src;
   error_status_t   status;
   #define          MINUTE 60
   #define          DAYSECS 86400

   /* ------------------- */
   /* infinite loop       */
   /* ------------------- */

   while (1)
   {

      /* ----------------------------------------- */
      /* wait till logon context is about to expire */
      /* ----------------------------------------- */

      sec_login_get_expiration (
         login_context,
         &expiration,
         &status);

      if ((status != rpc_s_ok) &&
          (status != sec_login_s_not_certified))
```

Figure 25 (Part 7 of 12). C Routine, SAVEPWS

```
{
   printf("Cannot get login context expiration time\n");
   fflush (NULL);
   exit(1);
}

current_time = time(NULL);
delay_time = expiration - current_time - (10*MINUTE);

if (delay_time > 0)
{
   delay.tv_sec = delay_time;
   delay.tv_nsec = 0;
   pthread_delay_np(&delay);
}

/* ---------------------------- */
/* refresh identity             */
/* ---------------------------- */

sec_login_refresh_identity(
   login_context,
   &status);

check_DCE_status(status,
   "Refreshing identity file");

/* ------------------------ */
/* Get Key From KEYTAB File */
/* ------------------------ */

sec_key_mgmt_get_key(
   rpc_c_authn_dce_secret,
   keytab,
   SERVER_PRINCIPAL_NAME,
   0,
   &server_key,
   &status);

check_DCE_status(status,
   "Getting Key from Keytab file");

/* ---------------------- */
/* Validate the Identity  */
/* ---------------------- */

identity_valid =
   sec_login_validate_identity(
      login_context,
      server_key,
      &reset_password,
      &auth_src,
      &status);

check_DCE_status(status,
   "Validating Identity");

/* ---------------- */
/* Free Key Storage */
/* ---------------- */

sec_key_mgmt_free_key(
   &server_key,
   &status);

check_DCE_status(status,
   "Freeing key storage");

/* ---------------- */
/* check refresh OK */
```

Figure 25 (Part 8 of 12). C Routine, SAVEPWS

```
      /* ---------------- */

      if (!identity_valid)
      {
        printf("Refresh of login context failed\n");
        fflush (NULL);
      }
   }
}

/*********************************************************************/
/*                                                                   */
/* Name:        check_DCE_status                                     */
/*                                                                   */
/* Description: check_DCE_status() implements status checking routine.*/
/*                                                                   */
/*********************************************************************/

void  check_DCE_status(error_status_t status, char *msg)
{

   /* ------------------- */
   /* local variables     */
   /* ------------------- */

   char            dce_error_string[256];
   error_status_t  error_inq_st;

   /* ------------------- */
   /* check status        */
   /* ------------------- */

   if ( status == error_status_ok )
   {
      if (DEBUG == 1)
      {
         printf("%s - OK\n", msg);
      }
   }
   else
   {
      printf("%s - ERROR\n", msg);

      dce_error_inq_text(
         status,
         dce_error_string,
         &error_inq_st);

      printf( "%s\n", dce_error_string);
      fflush(NULL);

      exit ( status );
   }
}

/*********************************************************************/
/*                                                                   */
/* Function:    svpw                                                 */
/*                                                                   */
/* Description: Save DCE password (encrypted) in user's RACF profile  */
/*                                                                   */
/*********************************************************************/

idl_long_int svpw(handle_t bh, char *password)
{

   /* ------------------------------- */
   /* local variables                 */
   /* ------------------------------- */
```

Figure 25 (Part 9 of 12). C Routine, SAVEPWS

```
int          logged_on;
idl_long_int saved_rc;
idl_long_int rc;
char         msg[128];
char         class[9];
char         entity[45];
char         access[20];
char         principal[129];
error_status_t   status;

/* ------------------------- */
/* logon client onto RACF    */
/* ------------------------- */

saved_rc = 0;
logged_on = 1;

rc = DCERACF(RACF_logon, bh);

if (rc != 0)
{
   logged_on = 0;
   saved_rc = rc;                /* save first error */
   RACF_logon_error(rc, msg);
   printf("%s - rc=%d\n", msg, rc);
   fflush(NULL);
   rc = rc | (RACF_logon << 16);
}

if (rc == 0)
{

   /* ------------------- */
   /* get DCE principal   */
   /* ------------------- */

   rc = DCERACF(RACF_get_DCE_principal, principal);

   /* ------------------------- */
   /* is this a stop request  */
   /* ------------------------- */

   if (strcmp(password, "-s-t-o-p-") == 0)
   {


      /* ------------------------------------------ */
      /* is the client authorised to stop the server? */
      /* ------------------------------------------ */

      strcpy(class,"$dceracf");
      strcpy(entity,"administrator");
      strcpy(access,"read");

      rc = DCERACF(RACF_check_authorisation,
                   bh,
                   class,
                   entity,
                   access);

      if (rc == 0)
      {
         printf("SAVEPW server has been requested to stop by %s\n",
                 principal);
         rpc_mgmt_stop_server_listening(NULL, &status);
         fflush(NULL);
         rc = 999;
         saved_rc = 999;
      }
      else
```

Figure 25 (Part 10 of 12). C Routine, SAVEPWS

```
         {
            if (saved_rc == 0)
            {
               saved_rc = rc | (RACF_check_authorisation << 16);
            }
            RACF_check_authorisation_error(rc, msg);
            printf("SAVEPW server has been requested to stop by %s, ",
                    principal);
            printf("%s - rc=%d\n", msg, rc);
            fflush(NULL);
            rc = rc | (RACF_check_authorisation << 16);
         }
      }
      else
      {

         /* -------------------- */
         /* set DCE password     */
         /* -------------------- */

         rc = DCERACF(RACF_set_DCE_password, password);

         if (rc == 0)
         {
            printf("Password changed for %s\n", principal);
         }
         else
         {
            if (saved_rc == 0)
            {
               saved_rc = rc;          /* save first error */
            }
            RACF_set_DCE_password_error(rc, msg);
            printf("%s - rc=%d\n", msg, rc);
            fflush(NULL);
            rc = rc | (RACF_set_DCE_password << 16);
         }
      }
   }

   /* ------------------------- */
   /* logoff RACF               */
   /* ------------------------- */

   if (logged_on == 1)
   {
      rc = DCERACF(RACF_logoff, bh);

      if (rc != 0)
      {
         if (saved_rc == 0)
         {
            saved_rc = rc;             /* save first error */
         }
         RACF_logoff_error(rc, msg);
         printf("%s - rc=%d\n", msg, rc);
         fflush(NULL);
         rc = rc | (RACF_logoff << 16);
      }
   }

   if (saved_rc != 0)
   {
     rc = saved_rc;                 /* restore first rc */
   }

   return(rc);

}
```

Figure 25 (Part 11 of 12). C Routine, SAVEPWS

Figure 25 (Part 12 of 12). C Routine, SAVEPWS

B.4 C Header Files

The data set hlq.DCERACF.H contains the following members:

DCERACF

PTKT

PTKTLIB

PTST

B.4.1 DCERACF

```
#ifndef DCERACF_common
#define DCERACF_common

   /*-------------------------------------------------------------------*/
   /*  DCERACF headers and functions                                    */
   /*                                                                   */
   /*    Note: These functions require the following #includes:         */
   /*                                                                   */
   /*        #include <dce/rpc.h>                                       */
   /*        #include <dce/sec_login.h>                                 */
   /*        #include <dce/keymgmt.h>                                   */
   /*        #include <dce/binding.h>                                   */
   /*        #include <dce/acct.h>                                      */
   /*        #include "dceracf.h"                                       */
   /*                                                                   */
   /*     This code uses the C tri graph characters for:                */
   /*                                                                   */
   /*         left  square bracket   ??(                                */
   /*         right square bracket   ??)                                */
   /*                                                                   */
   /*     to avoid possible character set complications on 3270         */
   /*     terminals/emulators.                                          */
   /*                                                                   */
   /*                                                                   */
   /*-------------------------------------------------------------------*/

   /*------------------------------------------------------------------ */
   /*  Defines                                                          */
   /*------------------------------------------------------------------ */

   #define RACF_logon                  1
   #define RACF_check_authorisation    2
   #define RACF_check_authorization    2
   #define RACF_logoff                 3
   #define RACF_set_DCE_password       4
   #define RACF_get_DCE_password       5
   #define RACF_get_DCE_principal      6
   #define RACF_set_DCE_principal      7
   #define RACF_get_DCE_usrdata        8
   #define RACF_get_MVS_sysid          9
   #define RACF_set_DCE_uuid          10
   #define RACF_get_DCE_uuid          11

   /*-------------------------------------------------------------------*/
   /*  Prototypes                                                       */
   /*-------------------------------------------------------------------*/

   int RACF_logon_error(int rc, char msg[]);
   int RACF_check_authorisation_error(int rc,char msg[]);
   int RACF_check_authorization_error(int rc,char msg[]);
   int RACF_logoff_error(int rc,char msg[]);
   int RACF_set_DCE_password_error(int rc,char msg[]);
   int RACF_get_DCE_password_error(int rc,char msg[]);
   int RACF_get_DCE_principal_error(int rc, char msg[]);
   int RACF_set_DCE_principal_error(int rc, char msg[]);
   int RACF_get_DCE_usrdata_error(int rc, char msg[]);
   int RACF_get_MVS_sysid_error(int rc, char msg[]);
   int RACF_set_DCE_uuid_error(int rc, char msg[]);
   int RACF_get_DCE_uuid_error(int rc, char msg[]);
   int dcelogin_using_RACF_profile(char principal[]);

   /*-------------------------------------------------------------------*/
   /*  Function: RACF_logon_error           (1)                         */
   /*-------------------------------------------------------------------*/
```

Figure 26 (Part 1 of 8). C Header File, DCERACF

```
int RACF_logon_error(int rc, char msg[])
{
   switch(rc)
   {
   case 4:
      strcpy(msg,"RACF userid has been revoked");
      break;
   case 8:
      strcpy(msg,"Invalid password or RACF userid not known to RACF");
      break;
   case 12:
      strcpy(msg, "DCE client is not using an authenicated RPC");
      break;
   case 16:
      strcpy(msg, "DCE client does not have a cross reference UUID");
      strcat(msg, " profile");
      break;
   case 20:
      strcpy(msg, "Requested function not recognized");
      break;
   default:
      strcpy(msg,"Unexpected return code");
   }
}

/*-------------------------------------------------------------------*/
/*  Function: RACF_check_authorisation_error           (2)         */
/*-------------------------------------------------------------------*/

int RACF_check_authorisation_error(int rc,char msg[])
{
   switch(rc)
   {
   case 8:
      strcpy(msg,"Access denied");
      break;
   case 12:
      strcpy(msg, "Resource is not known to RACF");
      break;
   case 16:
      strcpy(msg, "Class is not known to RACF");
      break;
   case 20:
      strcpy(msg, "Requested function not recognized");
      break;
   default:
      strcpy(msg,"Unexpected return code");
   }
}

/*-------------------------------------------------------------------*/
/*  Function: RACF_check_authorization_error           (2)         */
/*-------------------------------------------------------------------*/

int RACF_check_authorization_error(int rc,char msg[])
{
   switch(rc)
   {
   case 8:
      strcpy(msg,"Access denied");
      break;
   case 12:
      strcpy(msg, "Resource is not known to RACF");
      break;
   case 16:
      strcpy(msg, "Class is not known to RACF");
      break;
   case 20:
      strcpy(msg, "Requested function not recognized");
      break;
```

Figure 26 (Part 2 of 8). C Header File, DCERACF

```
   default:
      strcpy(msg,"Unexpected return code");
   }
}

/*-------------------------------------------------------------------*/
/*  Function: RACF_logoff_error                (3)                 */
/*-------------------------------------------------------------------*/

int RACF_logoff_error(int rc,char msg[])
{
   switch(rc)
   {
   case 8:
      strcpy(msg,"Userid not logged on");
      break;
   case 20:
      strcpy(msg, "Requested function not recognized");
      break;
   default:
      strcpy(msg,"Unexpected return code");
   }
}

/*-------------------------------------------------------------------*/
/*  Function: RACF_set_DCE_password_error      (4)                 */
/*-------------------------------------------------------------------*/

int RACF_set_DCE_password_error(int rc,char msg[])
{
   switch(rc)
   {
   case 8:
      strcpy(msg,"RACF user profile USRDATA field does not contain");
      strcat(msg,"DCE principal name");
      break;
   case 12:
      strcpy(msg, "RACF '<sysid>.CURRENT' profile not found");
      break;
   case 16:
      strcpy(msg, "Unable to encrypt pad");
      break;
   case 20:
      strcpy(msg, "Requested function not recognized");
      break;
   case 28:
      strcpy(msg, "Unable to update user profile");
      break;
   default:
      strcpy(msg,"Unexpected return code");
   }
}

/*-------------------------------------------------------------------*/
/*  Function: RACF_get_DCE_password_error     (5)                   */
/*-------------------------------------------------------------------*/

int RACF_get_DCE_password_error(int rc,char msg[])
{
   switch(rc)
   {
   case 8:
      strcpy(msg,"RACF user profile USRDATA field does not contain ");
      strcat(msg,"DCE password");
      break;
   case 12:
      strcpy(msg, "RACF '<sysid>.Vnnn' profile not found");
      break;
   case 16:
      strcpy(msg, "Unable to encrypt pad");
```

Figure 26 (Part 3 of 8). C Header File, DCERACF

```
      break;
   case 20:
      strcpy(msg, "Requested function not recognized");
      break;
   default:
      strcpy(msg,"Unexpected return code");
   }
}

/*-------------------------------------------------------------------*/
/*  RACF_get_DCE_principal_error          (6)                        */
/*-------------------------------------------------------------------*/

int RACF_get_DCE_principal_error(int rc, char msg[])
{
   switch(rc)
   {
   case 8:
      strcpy(msg, "DCE keyword not found in USRDATA");
      break;
   case 12:
      strcpy(msg, "DCE field in USRDATA is invalid");
      break;
   case 20:
      strcpy(msg, "Requested function not recognized");
      break;
   default:
      strcpy(msg,"Unexpected return code");
   }
}

/*-------------------------------------------------------------------*/
/*  RACF_set_DCE_principal_error     (7)                             */
/*-------------------------------------------------------------------*/

int RACF_set_DCE_principal_error(int rc, char msg[])
{
   switch(rc)
   {
   case 8:
      strcpy(msg, "DCE keyword not found in USRDATA");
      break;
   case 12:
      strcpy(msg, "DCE field in USRDATA is invalid");
      break;
   case 16:
      strcpy(msg, "Caller is not authorised to this request");
      break;
   case 20:
      strcpy(msg, "Requested function not recognized");
      break;
   default:
      strcpy(msg,"Unexpected return code");
   }
}

/*-------------------------------------------------------------------*/
/*  RACF_get_DCE_usrdata_error (8)                                   */
/*-------------------------------------------------------------------*/

int RACF_get_DCE_usrdata_error(int rc, char msg[])
{

   switch(rc)
   {
   case 12:
      strcpy(msg, "Userid not found");
      break;
   case 16:
      strcpy(msg, "Caller is not authorised to this request");
```

Figure 26 (Part 4 of 8). C Header File, DCERACF

```
      break;
   case 20:
      strcpy(msg, "Requested function not recognized");
      break;
   default:
      strcpy(msg,"Unexpected return code");
   }
}

/*-------------------------------------------------------------------*/
/*  RACF_get_MVS_sysid_error (9)                                     */
/*-------------------------------------------------------------------*/

int RACF_get_MVS_sysid_error(int rc, char msg[])
{

   switch(rc)
   {
   case 20:
      strcpy(msg, "Requested function not recognized");
      break;
   default:
      strcpy(msg,"Unexpected return code");
   }
}

/*-------------------------------------------------------------------*/
/*  RACF_set_DCE_uuid_error (10)                                     */
/*-------------------------------------------------------------------*/

int RACF_set_DCE_uuid_error(int rc, char msg[])
{

   switch(rc)
   {
   case 8:
      strcpy(msg, "Unable to create UUID profile");
      break;
   case 16:
      strcpy(msg, "You are not authorised to use this option");
      break;
   case 20:
      strcpy(msg, "Requested function not recognized");
      break;
   default:
      strcpy(msg,"Unexpected return code");
   }
}

/*-------------------------------------------------------------------*/
/*  RACF_get_DCE_uuid_error (11)                                     */
/*-------------------------------------------------------------------*/

int RACF_get_DCE_uuid_error(int rc, char msg[])
{

   switch(rc)
   {
   case 8:
      strcpy(msg, "UUID profile not found");
      break;
   case 20:
      strcpy(msg, "Requested function not recognized");
      break;
   default:
      strcpy(msg,"Unexpected return code");
   }
}

/* ------------------------------------------------------------------ */
```

Figure 26 (Part 5 of 8). C Header File, DCERACF

```
/*                                                                  */
/* Function: dcelogin_using_RACF_profile                            */
/*                                                                  */
/* Description: Login to DCE using the DCE principal and encrypted */
/*      DCE password held in the current RACF user profile          */
/*                                                                  */
/* ---------------------------------------------------------------- */

int dcelogin_using_RACF_profile(char principal[])
{
   /* ---------------------- */
   /* local variables        */
   /* ---------------------- */

   unsigned32   rc;
   unsigned32   i;
   boolean32    identity_valid;
   boolean32    reset_passwd;
   char         userid[9];
   char         password[129];
   char         old_password[129];
   char         new_password[129];
   char         verify_password[129];
   char         msg[129];
   char         registry_site[256];
   char         dce_error_string[256];
   sec_login_handle_t          login_context;
   sec_rgy_handle_t            registry_handle;
   error_status_t              error_inq_st;
   sec_passwd_rec_t            pwrec;
   sec_passwd_rec_t            caller_key;
   sec_passwd_version_t        new_password_version;
   sec_login_auth_src_t        auth_src;
   sec_rgy_bind_auth_info_t    registry_auth_info;
   sec_rgy_login_name_t        login_name;

   /* ---------------------- */
   /* get DCE principal name */
   /* ---------------------- */

   rc = DCERACF(RACF_get_DCE_principal, principal);

   if (rc != 0)
   {
      RACF_get_DCE_principal_error(rc, msg);
      printf("%s, rc=%d\n", msg, rc);
   }

   /* ---------------------- */
   /* get DCE password       */
   /* ---------------------- */

   if (rc == 0)
   {

      rc = DCERACF(RACF_get_DCE_password, password);

      if (rc != 0)
      {
         RACF_get_DCE_password_error(rc, msg);
         printf("%s, rc=%d\n", msg, rc);
         return(rc);
      }

      if (strlen(password) == 0)
      {
         rc = 1;
         return(rc);
      }
   }
```

Figure 26 (Part 6 of 8). C Header File, DCERACF

```
/* -------------------------------- */
/* setup identity in a login context */
/* -------------------------------- */

if (rc == 0)
{
   sec_login_setup_identity(
      principal,
      sec_login_no_flags,
      &login_context,
      &rc);

   if ( rc != error_status_ok ) {
      printf("Error setting identity\n");
      dce_error_inq_text(rc, dce_error_string, &error_inq_st);
      printf( "%s\n", dce_error_string);
      fflush(NULL);
   }
}

/* ------------------------ */
/* setup password record */
/* ------------------------ */

if (rc == 0)
{
   pwrec.key.tagged_union.plain = password;
   pwrec.key.key_type = sec_passwd_plain;
   pwrec.pepper = NULL;
   pwrec.version_number = sec_passwd_c_version_none;
}

/* ---------------------- */
/* Validate the Identity */
/* ---------------------- */

if (rc == 0)
{
   identity_valid = sec_login_validate_identity(
         login_context,
         &pwrec,
         &reset_passwd,
         &auth_src,
         &rc);

   if ( rc != error_status_ok ) {
      printf("Error validating identity\n");
      dce_error_inq_text(rc, dce_error_string, &error_inq_st);
      printf( "%s\n", dce_error_string);
      fflush(NULL);
   }
}

/* ----------------- */
/* Certify identity */
/* ----------------- */

if (rc == 0)
{
   sec_login_certify_identity(
      login_context,
      &rc);

   if ( rc != error_status_ok ) {
      printf("Error certifying identity\n");
      dce_error_inq_text(rc, dce_error_string, &error_inq_st);
      printf( "%s\n", dce_error_string);
      fflush(NULL);
   }
```

Figure 26 (Part 7 of 8). C Header File, DCERACF

```
      }

      /* ----------------- */
      /* Set login context */
      /* ----------------- */

      if (rc == 0)
      {
         sec_login_set_context(
            login_context,
            &rc);

         if ( rc != error_status_ok ) {
            printf("Error setting context\n");
            dce_error_inq_text(rc, dce_error_string, &error_inq_st);
            printf( "%s\n", dce_error_string);
            fflush(NULL);
         }
      }

      /* ------------------------------ */
      /* check login OK and authenicated */
      /* ------------------------------ */

      if (rc == 0)
      {
         if (identity_valid &&
             (auth_src == sec_login_auth_src_network))
         {
           rc = 0;
         }
         else
         {
           printf("Failed to login...\n");
           fflush (NULL);
           rc = 1;
         }
      }

      return(rc);
   }

#endif
```

Figure 26 (Part 8 of 8). C Header File, DCERACF

B.4.2 PTKT

```
/* Generated by IDL compiler version DCE 1.0.0-1 */
#ifndef PTKT_v1_0_included
#define PTKT_v1_0_included
#include <dce/idlbase.h>
#include <dce/dcerpcmsg.h>
#include <dce/rpc.h>

#ifdef __cplusplus
    extern "C" {
#endif

#include "dce/nbase.h"
#define MAX_CHAR (128)
typedef idl_char string_t[128];
extern idl_long_int passticket(
#ifdef IDL_PROTOTYPES
    /* [in] */ handle_t bh,
    /* [out] */ string_t ticket
#endif
);
typedef struct PTKT_v1_0_epv_t {
idl_long_int (*passticket)(
#ifdef IDL_PROTOTYPES
    /* [in] */ handle_t bh,
    /* [out] */ string_t ticket
#endif
);
} PTKT_v1_0_epv_t;
extern rpc_if_handle_t PTKT_v1_0_c_ifspec;
extern rpc_if_handle_t PTKT_v1_0_s_ifspec;

#ifdef __cplusplus
    }
#endif

#endif
```

Figure 27. C Header File, PTKT

B.4.3 PTKTLIB

```
/* ------------------------ */
/* included headers         */
/* ------------------------ */

#include "ptkt.h"

/* ------------------------ */
/* prototypes               */
/* ------------------------ */

void bind_to_passticket_server(
        handle_t *PTKT_bh,
        char     *sysid);
```

Figure 28. C Header File, PTKTLIB

B.4.4 PTST

```
/* Generated by IDL compiler version DCE 1.0.0-1 */
#ifndef PTST_v1_0_included
#define PTST_v1_0_included
#include <dce/idlbase.h>
#include <dce/dcerpcmsg.h>
#include <dce/rpc.h>

#ifdef __cplusplus
    extern "C" {
#endif

#include "dce/nbase.h"
#define MAX_CHAR (128)
typedef idl_char ptst_string_t[128];
extern idl_long_int ptst(
#ifdef IDL_PROTOTYPES
    /* [in] */ handle_t bh,
    /* [in] */ ptst_string_t ticket,
    /* [in] */ ptst_string_t dsn,
    /* [in] */ ptst_string_t record
#endif
);
typedef struct PTST_v1_0_epv_t {
idl_long_int (*ptst)(
#ifdef IDL_PROTOTYPES
    /* [in] */ handle_t bh,
    /* [in] */ ptst_string_t ticket,
    /* [in] */ ptst_string_t dsn,
    /* [in] */ ptst_string_t record
#endif
);
} PTST_v1_0_epv_t;
extern rpc_if_handle_t PTST_v1_0_c_ifspec;
extern rpc_if_handle_t PTST_v1_0_s_ifspec;

#ifdef __cplusplus
    }
#endif

#endif
```

Figure 29. C Header File, PTST

B.4.5 SVPW

```
/* Generated by IDL compiler version DCE 1.0.0-1 */
#ifndef SVPW_v1_0_included
#define SVPW_v1_0_included
#include <dce/idlbase.h>
#include <dce/dcerpcmsg.h>
#include <dce/rpc.h>

#ifdef __cplusplus
    extern "C" {
#endif

#include "dce/nbase.h"
#define MAX_CHAR (128)
typedef idl_char string_t[128];
extern idl_long_int svpw(
#ifdef IDL_PROTOTYPES
    /* [in] */ handle_t bh,
    /* [in] */ string_t password
#endif
);
typedef struct SVPW_v1_0_epv_t {
idl_long_int (*svpw)(
#ifdef IDL_PROTOTYPES
    /* [in] */ handle_t bh,
    /* [in] */ string_t password
#endif
);
} SVPW_v1_0_epv_t;
extern rpc_if_handle_t SVPW_v1_0_c_ifspec;
extern rpc_if_handle_t SVPW_v1_0_s_ifspec;

#ifdef __cplusplus
    }
#endif

#endif
```

Figure 30. C Header File, SVPW

B.5 ACF files

The data set hlq.DCERACF.ACF contains the following members:

PTKT

PTST

SVPW

B.5.1 PTKT

```
[explicit_handle] interface PTKT
{
}
```

Figure 31. ACF File, PTKT

B.5.2 PTST

```
[explicit_handle] interface PTST
{
}
```

Figure 32. ACF File, PTST

B.5.3 SVPW

```
[explicit_handle] interface SVPW
{
}
```

Figure 33. ACF File, SVPW

B.6 IDL Files

The data set hlq.DCERACF.IDL contains the following members:

PTKT

PTST

SVPW

B.6.1 PTKT

```
[
uuid(1a98a202-c1ca-1aa3-a036-00008c2ec018),
version(1.0)
]
interface PTKT
{
   const long MAX_CHAR   = 128;
   typedef [string] char string_t[MAX_CHAR];

   long int passticket(
      [in] handle_t bh,
      [out] string_t ticket
   );

}
```

Figure 34. IDL File, PTKT

B.6.2 PTST

```
[
uuid(ebb8f381-ac1c-1aa8-9006-00008c2ec018),
version(1.0)
]
interface PTST
{
   const long MAX_CHAR   = 128;
   typedef [string] char ptst_string_t[MAX_CHAR];

   long int ptst(
      [in] handle_t bh,
      [in] ptst_string_t ticket,
      [in] ptst_string_t dsn,
      [in] ptst_string_t record
   );

}
```

Figure 35. IDL File, PTST

B.6.3 SVPW

```
[
uuid(ca2db414-1c16-1a06-85ca-00008c2ec018),
version(1.0)
]
interface SVPW
{
   const long MAX_CHAR   = 128;
   typedef [string] char string_t[MAX_CHAR];

   long int svpw (
      [in] handle_t bh,
      [in] string_t password
   );

}
```

Figure 36. IDL File, SVPW

B.7 CLIST Files

The data set hlq.DCERACF.CLIST contains the following members:

DCERACFU

SAVEPW

B.7.1 DCERACFU

```
/* REXX */

arg p1 p2 p3 p4

"CALL 'hlq.DCERACF.LOAD(DCERACFU)' '"p1 p2 p3 p4"'"
```

Figure 37. CLIST, DCERACFU

B.7.2 SAVEPW

```
/* REXX */

arg p1 p2 p3 p4 p5

"CALL 'hlq.DCERACF.LOAD(SAVEPW)' '"p1 p2 p3 p4 p5"'"
```

Figure 38. CLIST, SAVEPW

B.8 JCL Files

The data set hlq.DCERACF.JCL contains the following members:

DCECC

DCEKEY

DCEKEYR

DCELK

DCERACF

DCERACFU

GETPAC

ICHRCX02

ICHRFR01

ICHRRCDE

PTKTGEN

PTKTIDL

PTKTLIB

PTKTS

PTKTSRUN

PTSTC

PTSTIDL

PTSTS

PTSTSRUN

SAVEPW

SAVEPWS

SAVEPWSR

SVPWIDL

USERSVC

B.8.1 DCECC

```
//*******************************************************************
//*                                                                 *
//* LICENSED MATERIALS - PROPERTY OF IBM                            *
//*                                                                 *
//* 5688-216 (C) COPYRIGHT IBM CORP. 1988, 1993                     *
//* ALL RIGHTS RESERVED                                             *
//*                                                                 *
//* US GOVERNMENT USERS RESTRICTED RIGHTS - USE,                    *
//* DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP                 *
//* SCHEDULE CONTRACT WITH IBM CORP                                 *
//*                                                                 *
//* SEE COPYRIGHT INSTRUCTIONS                                      *
//*                                                                 *
//*******************************************************************
//*                                                                 *
//* IBM SAA AD/CYCLE C/370                                          *
//*                                                                 *
//*  COMPILE A C PROGRAM                                            *
//*                                                                 *
//*  RELEASE LEVEL: 01.02.00  (VERSION.RELEASE.MODIFICATION LEVEL)  *
//*                                                                 *
//*  PARAMETER  DEFAULT VALUE    USAGE                              *
//*   CREGSIZ   4M               COMPILER REGION SIZE               *
//*   INFILE    NONE             INPUT DATA SET                     *
//*   OUTFILE   &&LOADSET        OUTPUT DATA SET                    *
//*   CPARM     NONE             COMPILER OPTIONS                   *
//*   DCB80     FB,80,3200       DCB FOR LRECL OF 80                *
//*   DCB3200   FB,3200,12800    DCB FOR LRECL OF 3200              *
//*   LIBPRFX   CEE.V1R3M0       PREFIX FOR LIBRARY DATA SET NAMES  *
//*   LNGPRFX   EDC.V1R2M0       PREFIX FOR LANGUAGE DATA SET NAMES *
//*   CMPPRFX   CEE.V1R3M0       COMPILER EXECUTABLE PREFIX         *
//*   DCEPRFX   DCE              PREFIX FOR DCE LIBRARY DATA SETS   *
//*                                                                 *
//*   SEOPT                      SEARCH CORRECTION  VANAERSCHOT     *
//*   DCEOPT                     DCE OPTIONS                        *
//*                                                                 *
//*******************************************************************
//*
//DCECC  PROC CREGSIZ='4M',
//   OUTFILE='&&LOADSET,DISP=(MOD,PASS),UNIT=VIO,SPACE=(TRK,(3,3))',
//   CPARM=,
//   SEOPT='SE(DD:USERLIB,DD:USERLIB2)',
//   DCEOPT='DEFINE(_OPEN_SYS,_DCE_THREADS,MVS)',
//   DCB80='(RECFM=FB,LRECL=80,BLKSIZE=3200)',
//   DCB3200='(RECFM=FB,LRECL=3200,BLKSIZE=12800)',
//   LIBPRFX='CEE.V1R3M0',
//   LNGPRFX='EDC.V1R2M0',
//   CMPPRFX='EDC.V1R2M0',
//   DCEPRFX='DCE'
//*
//*-----------------------------------------------------------------
//*  COMPILE STEP:
//*-----------------------------------------------------------------
//COMPILE EXEC PGM=EDCDC120,PARM=(,
//   '&CPARM,&SEOPT,&DCEOPT'),REGION=&CREGSIZ
//STEPLIB  DD  DSNAME=&CMPPRFX..SEDCDCMP,DISP=SHR
//         DD  DSNAME=&LIBPRFX..SCEERUN,DISP=SHR
//SYSMSGS  DD  DSNAME=&LNGPRFX..SEDCDMSG(EDCMSGE),DISP=SHR
//SYSIN    DD  DSNAME=&INFILE,DISP=SHR
//USERLIB  DD  DSNAME=&DCEPRFX..SEUVHDR,DISP=SHR
//SYSLIB   DD  DSNAME=&LNGPRFX..SEDCDHDR,DISP=SHR
//SYSLIN   DD  DSNAME=&OUTFILE
//SYSPRINT DD  SYSOUT=*
//SYSCPRT  DD  SYSOUT=*
//SYSUT1   DD  UNIT=VIO,SPACE=(32000,(30,30)),DCB=&DCB80
```

Figure 39 (Part 1 of 2). JCL Procedure, DCECC

```
//SYSUT4   DD  UNIT=VIO,SPACE=(32000,(30,30)),DCB=&DCB80
//SYSUT5   DD  UNIT=VIO,SPACE=(32000,(30,30)),DCB=&DCB3200
//SYSUT6   DD  UNIT=VIO,SPACE=(32000,(30,30)),DCB=&DCB3200
//SYSUT7   DD  UNIT=VIO,SPACE=(32000,(30,30)),DCB=&DCB3200
//SYSUT8   DD  UNIT=VIO,SPACE=(32000,(30,30)),DCB=&DCB3200
//SYSUT9   DD  UNIT=VIO,SPACE=(32000,(30,30)),
//             DCB=(RECFM=VB,LRECL=137,BLKSIZE=882)
//SYSUT10  DD  SYSOUT=*
//* PEND
```

Figure 39 (Part 2 of 2). JCL Procedure, DCECC

B.8.2 DCEKEY

```
//ERICFIN1 JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=A,REGION=5000K,
//    MSGLEVEL=(1,1)
//*--------------------------------------------------------------------
//*
//* NOTE: CHANGE 'hlq.DCERACF' TO '???.DCERACF' ALL
//*
//*--------------------------------------------------------------------
//*
//* ASSEMBLE DCEKEY
//* ---------------
//*
//DCEKEY   EXEC ASMHCL,PARM.C='SYSPARM(DEBUG),NODECK,OBJECT'
//C.SYSLIB  DD  DSN=hlq.DCERACF.ASM,DISP=SHR
//          DD  DSN=SYS1.MACLIB,DISP=SHR
//C.SYSPRINT DD SYSOUT=*
//C.SYSLIN  DD  DSN=hlq.DCERACF.OBJ(DCEKEY),DISP=SHR
//C.SYSIN   DD  DSN=hlq.DCERACF.ASM(DCEKEY),DISP=SHR
//L.SYSLMOD DD  DSN=SYS1.LINKLIB,DISP=SHR
//L.SYSLIN  DD  DSN=hlq.DCERACF.OBJ(DCEKEY),DISP=SHR
//          DD  *
    ENTRY DCEKEY
    SETCODE AC(1)
    NAME DCEKEY(R)
//*
```

Figure 40. JCL to Assemble and Link, DCEKEY

B.8.3 DCEKEYR

```
//ERICFIN1 JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=A,REGION=5000K,
//    MSGLEVEL=(1,1)
//*
//* CREATE A NEW SECRET SYSTEM ENCRYPTION KEY
//* -----------------------------------------
//*
//DCEKEYR  EXEC PGM=DCEKEY
//SYSUDUMP  DD  SYSOUT=*
```

Figure 41. JCL to Run, DCEKEYR

B.8.4 DCELK

```
//*******************************************************************
//*                                                                 *
//* LICENSED MATERIALS - PROPERTY OF IBM                            *
//*                                                                 *
//* 5688-198 (C) COPYRIGHT IBM CORP. 1992,1993                      *
//* ALL RIGHTS RESERVED                                             *
//*                                                                 *
//* US GOVERNMENT USERS RESTRICTED RIGHTS - USE,                    *
//* DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP                 *
//* SCHEDULE CONTRACT WITH IBM CORP                                 *
//*                                                                 *
//* SEE COPYRIGHT INSTRUCTIONS                                      *
//*                                                                 *
//*******************************************************************
//*                                                                 *
//* IBM SAA AD/CYCLE LANGUAGE ENVIRONMENT/370                       *
//*                                                                 *
//*  PRE-LINK AND LINK EDIT A C/370 PROGRAM                         *
//*                                                                 *
//*  RELEASE LEVEL: 01.03.00  (VERSION.RELEASE.MODIFICATION LEVEL)  *
//*                                                                 *
//*  PARAMETER  DEFAULT VALUE    USAGE                              *
//*   INFILE    NONE             INPUT DATA SET                     *
//*   OUTFILE   &&GSET(GO)       OUTPUT LINK EDIT DATA SET          *
//*   SYSLBLK   3200             BLOCKSIZE FOR &&PLKSET             *
//*   PREGSIZ   2048K            PRE-LINKER REGION SIZE             *
//*   PPARM     NONE             PRE-LINKER OPTIONS                 *
//*   PLIB      DUMMY            PRELINK AUTOCALL LIBRARY           *
//*   LIBPRFX   CEE.V1R3M0       PREFIX FOR LIBRARY DATA SET NAMES  *
//*   DCEPRFX   DCE              PREFIX FOR LIBRARY DATA SET NAMES  *
//*   LREGSIZ   1024K            LINK EDIT REGION SIZE              *
//*   LPARM     AMODE=31,MAP     LINK EDIT OPTIONS                  *
//*                                                                 *
//*******************************************************************
//*
//DCELK  PROC INFILE=,
// OUTFILE='&&GSET(GO),DISP=(MOD,PASS),UNIT=VIO,SPACE=(512,(50,20,1))',
//   SYSLBLK='3200',
//   PREGSIZ='2048K',
//   PPARM=,
//   PLIB='DCE.SEUVLIB',
//   LIBPRFX='CEE.V1R3M0',
//   DCEPRFX='DCE',
//   LREGSIZ='1024K',
//   LPARM='AMODE=31,MAP'
//*
//*-----------------------------------------------------------------
//* PRE-LINKEDIT STEP:
//*-----------------------------------------------------------------
//PLKED   EXEC PGM=EDCPRLK,PARM='&PPARM',
//    REGION=&PREGSIZ
//STEPLIB  DD  DSNAME=&LIBPRFX..SCEERUN,DISP=SHR
//SYSMSGS  DD  DSNAME=&LIBPRFX..SCEEMSGP(EDCPMSGE),DISP=SHR
//SYSLIB   DD  DSNAME=&PLIB,DISP=SHR
//SYSIN    DD  DSN=&INFILE,DISP=SHR
//SYSMOD   DD  DSNAME=&&PLKSET,UNIT=VIO,DISP=(MOD,PASS),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=&SYSLBLK)
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//*
//*-----------------------------------------------------------------
//* LINKEDIT STEP:
//*-----------------------------------------------------------------
//LKED   EXEC PGM=HEWL,COND=(4,LT,PLKED),
```

Figure 42 (Part 1 of 2). JCL Procedure, DCELK

```
//    REGION=&LREGSIZ,PARM='&LPARM'
//SYSLIB   DD  DSNAME=&LIBPRFX..SCEELKED,DISP=SHR
//         DD  DSNAME=&DCEPRFX..SEUVLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSNAME=*.PLKED.SYSMOD,DISP=SHR
//         DD  DDNAME=SYSIN
//SYSLMOD  DD  DSNAME=&OUTFILE,DISP=SHR
//SYSUT1   DD  UNIT=VIO,SPACE=(32000,(30,30))
//*
//* PEND
```

Figure 42 (Part 2 of 2). JCL Procedure, DCELK

B.8.5 DCERACF

```
//ERICFIN1 JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=5000K,
//    MSGLEVEL=(1,1)
//*--------------------------------------------------------------------
//*
//* ASSEMBLE DCERACF
//*
//* NOTE: CHANGE 'hlq.DCERACF' TO '???.DCERACF' ALL
//*
//*--------------------------------------------------------------------
//*
//DCERACF  EXEC ASMHC,PARM.C='SYSPARM(DEBUG),NODECK,OBJECT'
//C.SYSLIB  DD  DSN=hlq.DCERACF.ASM,DISP=SHR
//          DD  DSN=EDC.V2R1M0.SEDCMCLB,DISP=SHR
//          DD  DSN=SYS1.MACLIB,DISP=SHR
//C.SYSLIN  DD  DSN=hlq.DCERACF.OBJ(DCERACF),DISP=SHR
//C.SYSIN   DD  DSN=hlq.DCERACF.ASM(DCERACF),DISP=SHR
//*
```

Figure 43. JCL to Assemble, DCERACF

B.8.6 DCERACFU

```
//ERICFIN1 JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=5000K,
//    MSGLEVEL=(1,1)
//* --------------------------------------------------------------------
//*
//* NOTE: CHANGE 'hlq.DCERACF' TO '???.DCERACF' ALL
//*
//* --------------------------------------------------------------------
//*
//* COMPILE DCERACFU
//* ----------------
//*
//DCERACFU EXEC DCECC,
//  CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN   DD DSN=hlq.DCERACF.OBJ(DCERACFU),DISP=SHR
//COMPILE.USERLIB2 DD DSN=hlq.DCERACF.H,DISP=SHR
//COMPILE.SYSIN DD DSN=hlq.DCERACF.C(DCERACFU),DISP=SHR
//*
//* LINK DCERACFU
//* -------------
//*
//LKC       EXEC DCELK,OUTFILE='hlq.DCERACF.LOAD'
//PLKED.USERLIB    DD DSN=hlq.DCERACF.OBJ,DISP=SHR
//PLKED.SYSIN DD *
           INCLUDE USERLIB(DCERACFU)
//LKED.USERLIB  DD DSN=hlq.DCERACF.OBJ,DISP=SHR
//LKED.SYSPRINT DD SYSOUT=*
//LKED.SYSIN  DD *
           INCLUDE USERLIB(DCERACF)
           ENTRY CEESTART
           NAME    DCERACFU(R)
```

Figure 44. JCL to Compile and Link, DCERACFU

B.8.7 GETPAC

```
//ERICFIN1 JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=5000K,
//    MSGLEVEL=(1,1)
//* --------------------------------------------------------------------*
//*                                                                      *
//* NOTE: CHANGE 'hlq.DCERACF' TO '???.DCERACF' ALL                   *
//*                                                                      *
//* --------------------------------------------------------------------*
//*                                                                      *
//* COMPILE GETPAC                                                       *
//* --------------                                                       *
//*                                                                      *
//GETPAC   EXEC DCECC,CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN   DD DSN=hlq.DCERACF.OBJ(GETPAC),DISP=SHR
//COMPILE.USERLIB2 DD DSN=hlq.DCERACF.H,DISP=SHR
//COMPILE.SYSIN    DD DSN=hlq.DCERACF.C(GETPAC),DISP=SHR
```

Figure 45. JCL to Compile, GETPAC

B.8.8 ICHRCX02

```
//ERICFIN1 JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,                    00010000
//    CLASS=A,MSGCLASS=A,REGION=5000K,                                00020000
//    MSGLEVEL=(1,1)                                                  00030000
//*                                                                   00040000
//*--------------------------------------------------------------------*
//*
//* NOTE: CHANGE 'hlq.DCERACF' TO '???.DCERACF' ALL                 *
//*                                                                   *
//*--------------------------------------------------------------------*
//*                                                                   00040000
//STEP1 EXEC ASMHCL,                                                  00230002
//      PARM.C='OBJECT,NODECK',                                       00231002
//      PARM.L='RENT,REUS,LIST,MAP,LET,NCAL'                          00232002
//C.SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR                                00240001
//         DD DSN=SYS1.MODGEN,DISP=SHR                                00241001
//         DD DSN=hlq.DCERACF.ASM,DISP=SHR                            00242001
//C.SYSIN DD DSN=hlq.DCERACF.ASM(ICHRCX02),DISP=SHR                   00250000
//L.SYSLMOD DD DSN=SYS1.LPALIB,DISP=SHR                               00850000
//L.SYSIN   DD *                                                      00870000
      ENTRY   ICHRCX02                                              00871000
      SETCODE AC(1)                                                 00872000
      NAME    ICHRCX02(R)                                           00880000
/*                                                                  00890000
```

Figure 46. JCL to Assemble and Link, ICHRCX02

B.8.9 ICHRFR01

```
//ERICFIN1 JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,                      00010001
//    CLASS=A,MSGCLASS=A,REGION=5000K,                                  00020001
//    MSGLEVEL=(1,1)                                                    00030001
//*                                                                     00040001
//*******************************************************************//  08850000
//*  RFTABLE:                                                       *//  08900000
//*                                                                 *//  08950000
//*      CONTINUATION OF THE RACF INSTALLATION PROCEDURE STEP: @03C*//  09029901
//*      UPDATE THE RACF ROUTER TABLE.                          @03C*//  09059901
//*                                                                 *//  09100000
//*      THIS STEP IS OPTIONAL FOR EXISTING AND NEW INSTALLATIONS  *//  09150000
//*                                                                 *//  09200000
//*      THIS MEMBER CONTAINS A JOB WHICH CAN BE USED TO ASSEMBLE  *//  09250000
//*      AND LINK EDIT NEW TABLE ENTRIES TO THE OPTIONAL           *//  09300000
//*      INSTALLATION DEFINED RACF ROUTER TABLE.                   *//  09350000
//*                                                                 *//  09400000
//*      MODIFY THIS SAMPLE TO YOUR INSTALLATIONS NEEDS BEFORE     *//  09450000
//*      USING IT.  SEE THE INSTRUCTIONS BELOW ON HOW TO MODIFY    *//  09500000
//*      THIS JOB.                                                 *//  09550000
//*                                                                 *//  09600000
//*******************************************************************//  09650000
//*                                        STEP1 and C.SYSLIB  @03C*//  09719901
//STEP1     EXEC ASMHCL,PARM.C='OBJECT,NODECK'                          09739901
//C.SYSLIB DD DSN=SYS1.MODGEN,DISP=SHR                                  09759901
//C.SYSIN DD *                                                          09966600
**********************************************************************  10000000
**********************************************************************  10050000
*                                                                  *  10100000
*              SAMPLE RACF ROUTER TABLE JOB                        *  10150000
*                                                                  *  10200000
*   THIS IS A SAMPLE RACF ROUTER TABLE WHICH PROVIDES ACTION CODES *  10250000
*   FOR INSTALLATION-DEFINED ROUTER TABLE ENTRIES,                 *  10300000
*   TO DETERMINE WHETHER OR NOT RACF IS INVOKED ON BEHALF OF THE   *  10350000
*   RACROUTE MACRO.                                                *  10400000
*                                                                  *  10450000
*   THIS SAMPLE RACF ROUTER TABLE CAN BE USED TO                   *  10500000
*   ADD, MODIFY OR DELETE ENTRIES AS NEEDED.                       *  10550000
*   NOTE THAT THE ICHRFRTB MACRO NEEDED TO ASSEMBLE THE            *  10600000
*   ICHRFR01 MODULE IS IN SYS1.MODGEN.                             *  10650000
*                                                                  *  10700000
*   TO ACCOMPLISH THIS, USE ONE OF THE FOLLOWING METHODS           *  10750000
*   (THIS JOB IS A SAMPLE FOR THE FIRST METHOD):                   *  10800000
*                                                                  *  10850000
*         - THE RACF ROUTER MACRO, ICHRFRTB, WHICH                 *  10900000
*           HAS BEEN PLACED IN SYS1.MODGEN, IS TO BE INVOKED       *  10950000
*           FOR EACH ENTRY. ASSEMBLE AND LINK                      *  11000000
*           EDIT THE RESULT TO PRODUCE THE LOAD MODULE             *  11050000
*           USED BY RACF.                                          *  11100000
*                                                                  *  11150000
*         - LINKEDIT THE OBJECT MODULE FOR THE ENTRIES             *  11200000
*           BEING ADDED OR MODIFIED TOGETHER WITH AN               *  11250000
*           EXISTING LOAD MODULE ICHRFR01 IN SYS1.LINKLIB          *  11300000
*           TO PRODUCE A NEW LOAD MODULE.  AN ENTRY CAN BE         *  11350000
*           DELETED FROM THE LIST BY SPECIFYING                    *  11400000
*           THE NAME OF THE ENTRY TO BE DELETED USING THE          *  11450000
*           LINKAGE EDITOR REPLACE STATEMENT.  WHEN YOU USE        *  11500000
*           THIS METHOD, YOU CAN ADD OBJECT FOR NEW CLASSES        *  11550000
*           TO THE LOAD MODULE WITHOUT REASSEMBLING.               *  11600000
*                                                                  *  11650000
*                                                                  *  11700000
*                        N O T E                                   *  11750000
*                                                                  *  11800000
*   FOR RACF VERSION 1 RELEASE 7 AND HIGHER, THE IBM-DEFINED RFR   *  11850000
*   ENTRIES ARE CONTAINED IN THE ICHRFR01 LOAD MODULE SUPPLIED WITH*  11900000
*   THIS PRODUCT.                                                  *  11950000
```

Figure 47 (Part 1 of 2). JCL to Assemble and Link, ICHRFR01

```
*                                                                     *  12000000
*   FOR RACF VERSION 1 RELEASE 7 AND HIGHER, THE ICHRFR01 LOAD        *  12050000
*   MODULE SHOULD BE PLACED IN SYS1.LINKLIB.                          *  12100000
*                                                                     *  12150000
*   FOR FURTHER INFORMATION CONCERNING THE ICHRFR01 MODULE AND        *  12200000
*   HOW TO CODE THE ICHRFRTB MACRO, PLEASE REFER TO THE "RACF MACROS *  12250000
*   AND INTERFACES" PUBLICATION.                                      *  12300000
*                                                                     *  12350000
*   THE FOLLOWING IS A SAMPLE INSTALLATION-DEFINED ROUTER TABLE:      *  12400000
*                                                                     *  12450000
*                                                                     *  12500000
*********************************************************************  12550000
*********************************************************************  12600000
*                                                                     *  12650000
ICHRFR01 CSECT                                                          12700000
$DCERACF ICHRFRTB CLASS=$DCERACF,ACTION=RACF                            12750002
ENDTAB   ICHRFRTB TYPE=END                                              12800000
         END   ICHRFR01                                                 12850000
/*                                                                      12900000
//L.SYSLMOD DD DSN=SYS1.LINKLIB,                                        12950001
//             DISP=SHR                                                 13000001
//L.SYSIN   DD *                                                        13100000
      NAME    ICHRFR01(R)                                             13150000
/*                                                                      13200000
```

Figure 47 (Part 2 of 2). JCL to Assemble and Link, ICHRFR01

B.8.10 ICHRRCDE

```
//ERICFINR JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,                      00010000
//    CLASS=A,MSGCLASS=A,REGION=5000K,                                  00020000
//    MSGLEVEL=(1,1)                                                    00030000
//*                                                                     00040000
//*******************************************************************// 03600000
//*  CDTABLE:                                                       *// 03650000
//*                                                                 *// 03700000
//*      RACF INSTALLATION PROCEDURE STEP: UPDATE             @O3C*// 03769900
//*      THE RACF CLASS DESCRIPTOR TABLE.                     @P2C*// 03800000
//*                                                                 *// 03850000
//*      THIS STEP IS OPTIONAL FOR EXISTING AND NEW INSTALLATIONS  *// 03900000
//*                                                                 *// 03950000
//*      THIS IS SAMPLE JOB WHICH CAN BE USED TO ASSEMBLE          *// 04000000
//*      AND LINK EDIT NEW CLASSES TO THE CLASS DESCRIPTOR TABLE,  *// 04050000
//*      ICHERCDE, CONTAINING INSTALLATION-DEFINED CLASSES.        *// 04100000
//*                                                                 *// 04150000
//*      MODIFY THIS SAMPLE TO YOUR INSTALLATIONS NEEDS BEFORE     *// 04200000
//*      USING IT.  SEE THE INSTRUCTIONS BELOW ON HOW TO MODIFY    *// 04250000
//*      THIS JOB.                                                 *// 04300000
//*                                                                 *// 04350000
//*******************************************************************// 04400000
//*                                        STEP1 and C.SYSLIB  @O3C*// 04469900
//STEP1     EXEC ASMHCL,PARM.C='OBJECT,NODECK'                        04489900
//C.SYSLIB DD DSN=SYS1.MODGEN,DISP=SHR                                04509900
//C.SYSIN DD *                                                        04716600
********************************************************************** 04750000
********************************************************************** 04800000
*                                                                  *  04850000
*              SAMPLE CLASS DESCRIPTOR JOB                         *  04900000
*                                                                  *  04950000
*   THIS IS A SAMPLE CLASS DESCRIPTOR TABLE WHICH REPRESENTS       *  05000000
*   INSTALLATION-DEFINED RESOURCE CLASSES.  EACH                   *  05050000
*   CLASS DESCRIPTOR ENTRY CONTAINS CONTROL INFORMATION NEEDED     *  05100000
*   BY RACF TO VALIDATE CLASS NAMES AND IS A CSECT IN THE          *  05150000
*   LOAD MODULE ICHRRCDE.  THE LAST CSECT IN THE LOAD              *  05200000
*   MODULE IS ICHRRCDE AND INDICATES THE END OF THE TABLE.         *  05250000
*                                                                  *  05300000
*   THIS SAMPLE CLASS DESCRIPTOR TABLE CAN BE USED TO              *  05350000
*   ADD, MODIFY OR DELETE INSTALLATION-DEFINED                     *  05400000
*   CLASS DESCRIPTOR ENTRIES AS NEEDED.                            *  05450000
*   NOTE THAT THE ICHERCDE MACRO NEEDED TO ASSEMBLE THE            *  05500000
*   ICHRRCDE MODULE IS IN SYS1.MODGEN.                             *  05550000
*                                                                  *  05600000
*   TO ACCOMPLISH THIS, USE ONE OF THE FOLLOWING METHODS           *  05650000
*   (THIS JOB IS A SAMPLE FOR THE FIRST METHOD):                   *  05700000
*                                                                  *  05750000
*         - INVOKE THE CLASS DESCRIPTOR MACRO, ICHERCDE,           *  05800000
*           WHICH HAS BEEN PLACED IN SYS1.MODGEN, FOR              *  05850000
*           EACH RESOURCE CLASS AND ASSEMBLE.  LINK                *  05900000
*           EDIT THE RESULT TO PRODUCE THE LOAD MODULE             *  05950000
*           USED BY RACF.  WHEN YOU USE THIS METHOD,               *  06000000
*           THE ICHERCDE MACRO CROSS-CHECKS CLASS                  *  06050000
*           DESCRIPTORS TO ENSURE THAT NO ERRORS EXIST             *  06100000
*           (I.E., THE FIRST FOUR CHARACTERS OF CLASS              *  06150000
*           NAMES ARE UNIQUE).                                     *  06200000
*                                                                  *  06250000
*         - LINKEDIT THE OBJECT MODULE(S) FOR THE CLASS(ES)        *  06300000
*           BEING ADDED OR MODIFIED TOGETHER WITH ANY              *  06350000
*           EXISTING LOAD MODULE ICHRRCDE IN SYS1.LINKLIB          *  06400000
*           TO PRODUCE A NEW LOAD MODULE.  A CLASS CAN BE          *  06450000
*           DELETED FROM THE DESCRIPTOR LIST BY SPECIFYING         *  06500000
*           THE NAME OF THE CLASS TO BE DELETED USING THE          *  06550000
*           LINKAGE EDITOR REPLACE STATEMENT.  WHEN YOU USE        *  06600000
*           THIS METHOD, YOU CAN ADD OBJECT FOR NEW CLASSES        *  06650000
*           TO THE LOAD MODULE WITHOUT REASSEMBLING.               *  06700000
```

Figure 48 (Part 1 of 2). JCL to Assemble and Link, ICHRRCDE

```
*      ++++++THE LAST CSECT IN THE LOAD MUST BE ICHRRCDE+++++        *  06750000
*                                                                   *  06800000
*                                                                   *  06850000
*                         N O T E                                   *  06900000
*                                                                   *  06950000
*   FOR RACF VERSION 1 RELEASE 7 AND HIGHER, THE IBM-DEFINED CDT    *  07000000
*   ENTRIES ARE CONTAINED IN THE ICHRRCDX LOAD MODULE SUPPLIED WITH  *  07050000
*   THIS PRODUCT.                                                   *  07100000
*                                                                   *  07150000
*   FOR RACF VERSION 1 RELEASE 7 AND HIGHER, THE ICHRRCDE LOAD      *  07200000
*   MODULE SHOULD BE PLACED IN SYS1.LINKLIB.                        *  07250000
*                                                                   *  07300000
*   FOR FURTHER INFORMATION CONCERNING THE ICHRRCDE MODULE AND      *  07350000
*   HOW TO CODE THE ICHERCDE MACRO, PLEASE REFER TO THE "RACF MACROS *  07400000
*   AND INTERFACES" PUBLICATION.                                    *  07450000
*                                                                   *  07500000
*   THE FOLLOWING IS A SAMPLE INSTALLATION-DEFINED CDT:             *  07550000
*                                                                   *  07600000
*********************************************************************  07650000
*********************************************************************  07700000
$DCERACF ICHERCDE CLASS=$DCERACF,                                     +08060000
               ID=128,                                                +08070000
               MAXLNTH=64,                                            +08080000
               FIRST=ALPHANUM,                                        +08090000
               OTHER=ANY,                                             +08091000
               POSIT=25,                                              +08092000
               DFTUACC=NONE,                                          +08093000
               OPER=NO                                                 08094000
         ICHERCDE                                                      08100000
/*                                                                     08150000
//L.SYSLMOD DD DSN=SYS1.LINKLIB,DISP=SHR,VOL=SER=EPORS1,UNIT=3390      08200000
//L.SYSIN   DD *                                                       08350000
      ORDER   $DCERACF                                                08560000
      ORDER   ICHRRCDE                                                08600000
      NAME    ICHRRCDE(R)                                             08650000
/*                                                                     08700000
```

Figure 48 (Part 2 of 2). JCL to Assemble and Link, ICHRRCDE

B.8.11 PTKTGEN

```
//ERICFIN1 JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=5000K,
//    MSGLEVEL=(1,1)
//*--------------------------------------------------------------------
//*
//* ASSEMBLE PTKTGEN
//*
//* NOTE: CHANGE 'ERICFIN.DCE' TO '<HLQ>.DCERACF' ALL
//*
//*--------------------------------------------------------------------
//*
//PTKTGEN  EXEC ASMHC,PARM.C='SYSPARM(DEBUG),NODECK,OBJECT'
//C.SYSLIB  DD  DSN=ERICFIN.DCE.ASM,DISP=SHR
//          DD  DSN=EDC.V2R1M0.SEDCMCLB,DISP=SHR
//          DD  DSN=SYS1.MACLIB,DISP=SHR
//          DD  DSN=SYS1.MODGEN,DISP=SHR
//C.SYSLIN  DD  DSN=ERICFIN.DCE.OBJ(PTKTGEN),DISP=SHR
//C.SYSIN   DD  DSN=ERICFIN.DCE.ASM(PTKTGEN),DISP=SHR
//*
```

Figure 49. JCL to Assemble, PTKTGEN

B.8.12 PTKTIDL

```
//ERICFINI JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=8000K,
//    MSGLEVEL=(1,1)
//NEWLK    JCLLIB ORDER=(DCE.SEUVPRC)
//* --------------------------------------------------------------------
//*
//* NOTE: CHANGE 'ERICFIN.DCE' TO '<HLQ>.DCERACF' ALL                  *
//*                                                                    *
//* --------------------------------------------------------------------
//*                                                                    *
//* GENERATE THE STUBS AND IDL HEADER
//* ---------------------------------
//*
//IDLCOMP  EXEC IDL,USERPFX='ERICFIN.DCE',DCEPFX='DCE',REGSIZE=8000K,
//    PARMS='PTKT -v -client all -server all -keep all'
//
```

Figure 50. JCL to IDL Compile, PTKTIDL

B.8.13 PTKTLIB

```
//ERICFIN1 JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=5000K,
//    MSGLEVEL=(1,1)
//* --------------------------------------------------------------------*
//*                                                                    *
//* NOTE: CHANGE 'ERICFIN.DCE' TO '<HLQ>.DCERACF' ALL                  *
//*                                                                    *
//* --------------------------------------------------------------------*
//*                                                                    *
//* COMPILE PTKTLIB                                                    *
//* --------------                                                     *
//*                                                                    *
//PTKTLIB  EXEC DCECC,CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN   DD DSN=ERICFIN.DCE.OBJ(PTKTLIB),DISP=SHR
//COMPILE.USERLIB2 DD DSN=ERICFIN.DCE.H,DISP=SHR
//COMPILE.SYSIN    DD DSN=ERICFIN.DCE.C(PTKTLIB),DISP=SHR
```

Figure 51. JCL to Compile, PTKTLIB

B.8.14 PTKTS

```
//ERICFINS JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
// CLASS=A,MSGCLASS=H,REGION=5000K,
// MSGLEVEL=(1,1)
//* ------------------------------------------------------------------
//*
//* NOTE: CHANGE 'ERICFIN.DCE' TO '<HLQ>.DCERACF' ALL
//*
//* ------------------------------------------------------------------
//*
//* COMPILE THE PTKT SERVER
//* -----------------------
//*
//PTKTS EXEC DCECC,
// CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN DD DSN=ERICFIN.DCE.OBJ(PTKTS),DISP=SHR
//COMPILE.USERLIB2 DD DSN=ERICFIN.DCE.H,DISP=SHR
//COMPILE.SYSIN DD DSN=ERICFIN.DCE.C(PTKTS),DISP=SHR
//*
//* COMPILE THE PTKT SERVER STUB
//* ----------------------------
//*
//PTKTSS EXEC DCECC,
// CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN DD DSN=ERICFIN.DCE.OBJ(PTKTSS),DISP=SHR
//COMPILE.USERLIB2 DD DSN=ERICFIN.DCE.H,DISP=SHR
//COMPILE.SYSIN DD DSN=ERICFIN.DCE.C(PTKTSS),DISP=SHR
//*
//* LINK THE PTKT SERVER
//* --------------------
//*
//LINK      EXEC DCELK,OUTFILE='SYS1.LINKLIB'
//USERLIB DD DSN=ERICFIN.DCE.OBJ,DISP=SHR
//PLKED.SYSIN DD *
        INCLUDE USERLIB(GETPAC)
        INCLUDE USERLIB(PTKTSS)
        INCLUDE USERLIB(PTKTS)
//LKED.USERLIB DD DSN=ERICFIN.DCE.OBJ,DISP=SHR
//LKED.SYSPRINT DD SYSOUT=*
//LKED.SYSIN DD *
        SETCODE AC(1)
        INCLUDE USERLIB(DCERACF)
        INCLUDE USERLIB(PTKTGEN)
        NAME    PTKTS(R)
//*
```

Figure 52. JCL to Compile and Link, PTKTS

B.8.15 PTKTSRUN

```
//ERICFINP JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=64M,TIME=1440,
//    MSGLEVEL=(1,1)
//* ------------------------------------------------------------------*
//*                                                                   *
//* Note: CHANGE 'ERICFIN.DCE' TO '<HLQ>.DCERACF' ALL                 *
//*                                                                   *
//* Note: The PARM is the HFS KEYTAB file that has the 'passticket'   *
//*       principal entry.                                            *
//*                                                                   *
//* Note: EUVSJRB5 DD points to the ticket cache to be used.          *
//*                                                                   *
//* ------------------------------------------------------------------*
//*                                                                   *
//* EXECUTE THE PTKT SERVER                                           *
//* ------------------------                                          *
//*                                                                   *
//PTKTS EXEC PGM=PTKTS,REGION=64M,
//             PARM='//u/ericfin/passticket.key'
//EUVSKRB5 DD PATH='/u/ericfin/krb5ccname.passticket'
//SYSPRINT DD SYSOUT=*
//*
```

Figure 53. JCL to Run, PTKTSRUN

B.8.16 PTSTC

```
//ERICFINC JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=5000K,
//    MSGLEVEL=(1,1)
//* ------------------------------------------------------------------
//*
//* NOTE: CHANGE 'ERICFIN.DCE' TO '<HLQ>.DCERACF' ALL
//*
//* ------------------------------------------------------------------
//*
//* COMPILE THE PTSTC CLIENT
//* -------------------------
//*
//PTSTC    EXEC DCECC,
//  CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN   DD DSN=ERICFIN.DCE.OBJ(PTSTC),DISP=SHR
//COMPILE.USERLIB2 DD DSN=ERICFIN.DCE.H,DISP=SHR
//COMPILE.SYSIN DD DSN=ERICFIN.DCE.C(PTSTC),DISP=SHR
//*
//* COMPILE THE PTSTC  STUB.
//* ------------------------
//*
//PTSTCS   EXEC DCECC,
//  CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN   DD DSN=ERICFIN.DCE.OBJ(PTSTCS),DISP=SHR
//COMPILE.USERLIB2 DD DSN=ERICFIN.DCE.H,DISP=SHR
//COMPILE.SYSIN DD DSN=ERICFIN.DCE.C(PTSTCS),DISP=SHR
//*
//* COMPILE THE PTKTC  STUB.
//* ------------------------
//*
//PTKTCS   EXEC DCECC,
//  CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN   DD DSN=ERICFIN.DCE.OBJ(PTKTCS),DISP=SHR
//COMPILE.USERLIB2 DD DSN=ERICFIN.DCE.H,DISP=SHR
//COMPILE.SYSIN DD DSN=ERICFIN.DCE.C(PTKTCS),DISP=SHR
//*                                                                  *
//* COMPILE PTKTLIB                                                  *
//* --------------                                                   *
//*                                                                  *
//PTKTLIB  EXEC DCECC,CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN   DD DSN=ERICFIN.DCE.OBJ(PTKTLIB),DISP=SHR
//COMPILE.USERLIB2 DD DSN=ERICFIN.DCE.H,DISP=SHR
//COMPILE.SYSIN    DD DSN=ERICFIN.DCE.C(PTKTLIB),DISP=SHR
//*
//* LINK THE CLIENT
//* ---------------
//*
//LINK      EXEC DCELK,OUTFILE='ERICFIN.DCE.LOAD'
//PLKED.USERLIB    DD DSN=ERICFIN.DCE.OBJ,DISP=SHR
//PLKED.SYSIN DD *
           INCLUDE USERLIB(PTKTLIB)
           INCLUDE USERLIB(PTKTCS)
           INCLUDE USERLIB(PTSTCS)
           INCLUDE USERLIB(PTSTC)
//LKED.USERLIB  DD DSN=ERICFIN.DCE.OBJ,DISP=SHR
//LKED.SYSPRINT DD SYSOUT=*
//LKED.SYSIN  DD *
           NAME    PTSTC(R)
//*
```

Figure 54. JCL to Compile and Link, PTSTC

B.8.17 PTSTIDL

```
//ERICFINI JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=8000K,
//    MSGLEVEL=(1,1)
//NEWLK    JCLLIB ORDER=(DCE.SEUVPRC)
//* ------------------------------------------------------------------
//*
//* NOTE: CHANGE 'ERICFIN.DCE' TO '<HLQ>.DCERACF' ALL                 *
//*                                                                   *
//* ------------------------------------------------------------------
//*                                                                   *
//* GENERATE THE STUBS AND IDL HEADER
//* ---------------------------------
//*
//IDLCOMP  EXEC IDL,USERPFX='ERICFIN.DCE',DCEPFX='DCE',REGSIZE=8000K,
//    PARMS='PTST -v -client all -server all -keep all'
//
```

Figure 55. JCL to IDL Compile, PTSTIDL

B.8.18 PTSTS

```
//ERICFINS JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
// CLASS=A,MSGCLASS=H,REGION=5000K,
// MSGLEVEL=(1,1)
//* ------------------------------------------------------------------
//*
//* NOTE: CHANGE 'ERICFIN.DCE' TO '<HLQ>.DCERACF' ALL
//*
//* ------------------------------------------------------------------
//*
//* COMPILE THE PTST SERVER
//* -----------------------
//*
//PTSTS    EXEC DCECC,
// CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN DD DSN=ERICFIN.DCE.OBJ(PTSTS),DISP=SHR
//COMPILE.USERLIB2 DD DSN=ERICFIN.DCE.H,DISP=SHR
//COMPILE.SYSIN DD DSN=ERICFIN.DCE.C(PTSTS),DISP=SHR
//*
//* COMPILE THE PTST SERVER STUB
//* ----------------------------
//*
//PTSTSS EXEC DCECC,
// CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN DD DSN=ERICFIN.DCE.OBJ(PTSTSS),DISP=SHR
//COMPILE.USERLIB2 DD DSN=ERICFIN.DCE.H,DISP=SHR
//COMPILE.SYSIN DD DSN=ERICFIN.DCE.C(PTSTSS),DISP=SHR
//*
//* LINK THE PTST SERVER
//* --------------------
//*
//LINK     EXEC DCELK,OUTFILE='ERICFIN.DCE.LOAD'
//USERLIB DD DSN=ERICFIN.DCE.OBJ,DISP=SHR
//PLKED.SYSIN DD *
        INCLUDE USERLIB(PTSTSS)
        INCLUDE USERLIB(PTSTS)
//LKED.USERLIB DD DSN=ERICFIN.DCE.OBJ,DISP=SHR
//LKED.SYSPRINT DD SYSOUT=*
//LKED.SYSIN DD *
        INCLUDE USERLIB(DCERACF)
        NAME    PTSTS(R)
//*
```

Figure 56. JCL to Compile and Link, PTSTS

B.8.19 PTSTSRUN

```
//ERICFINT JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=64M,TIME=1440,
//    MSGLEVEL=(1,1)
//* --------------------------------------------------------------------*
//*                                                                     *
//* Note: CHANGE 'ERICFIN.DCE' TO '<HLQ>.DCERACF' ALL                   *
//*                                                                     *
//* Note: The PARM is the HFS KEYTAB file that has the 'passticket'     *
//*       principal entry.                                              *
//*                                                                     *
//* Note: EUVSJRB5 DD points to the ticket cache to be used.            *
//*                                                                     *
//* --------------------------------------------------------------------*
//*                                                                     *
//* EXECUTE THE PTST SERVER                                             *
//* ------------------------                                            *
//*                                                                     *
//PTSTS   EXEC PGM=PTSTS,REGION=64M,
//             PARM='//u/ericfin/ptst.key'
//STEPLIB DD DSN=ERICFIN.DCE.LOAD,DISP=SHR
//EUVSKRB5 DD PATH='/u/ericfin/krb5ccname.ptst'
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//*
```

Figure 57. JCL to Run, PTSTSRUN

B.8.20 SAVEPW

```
//ERICFINC JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=5000K,
//    MSGLEVEL=(1,1)
//* ------------------------------------------------------------------
//*
//* NOTE: CHANGE 'hlq.DCERACF' TO '???.DCERACF' ALL
//*
//* ------------------------------------------------------------------
//*
//* COMPILE THE SAVEPW CLIENT
//* -------------------------
//*
//SAVEPW   EXEC DCECC,
//  CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN   DD DSN=hlq.DCERACF.OBJ(SAVEPW),DISP=SHR
//COMPILE.USERLIB2 DD DSN=hlq.DCERACF.H,DISP=SHR
//COMPILE.SYSIN DD DSN=hlq.DCERACF.C(SAVEPW),DISP=SHR
//*
//* COMPILE THE CLIENT STUB.
//* ------------------------
//*
//SVPWCS   EXEC DCECC,
//  CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN   DD DSN=hlq.DCERACF.OBJ(SVPWCS),DISP=SHR
//COMPILE.USERLIB2 DD DSN=hlq.DCERACF.H,DISP=SHR
//COMPILE.SYSIN DD DSN=hlq.DCERACF.C(SVPWCS),DISP=SHR
//*
//* LINK THE CLIENT
//* ---------------
//*
//LINK      EXEC DCELK,OUTFILE='hlq.DCERACF.LOAD'
//PLKED.USERLIB    DD DSN=hlq.DCERACF.OBJ,DISP=SHR
//PLKED.SYSIN DD *
           INCLUDE USERLIB(SVPWCS)
           INCLUDE USERLIB(SAVEPW)
//LKED.USERLIB  DD DSN=hlq.DCERACF.OBJ,DISP=SHR
//LKED.SYSPRINT DD SYSOUT=*
//LKED.SYSIN  DD *
           INCLUDE USERLIB(DCERACF)
           NAME    SAVEPW(R)
//*
```

Figure 58. JCL to Compile and Link, SAVEPW

B.8.21 SAVEPWS

```
//ERICFINS JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
// CLASS=A,MSGCLASS=H,REGION=5000K,
// MSGLEVEL=(1,1)
//* ------------------------------------------------------------------
//*
//* NOTE: CHANGE 'hlq.DCERACF' TO '???.DCERACF' ALL
//*
//* ------------------------------------------------------------------
//*
//* COMPILE THE SAVEPW SERVER
//* -------------------------
//*
//SAVEPWS EXEC DCECC,
// CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN DD DSN=hlq.DCERACF.OBJ(SAVEPWS),DISP=SHR
//COMPILE.USERLIB2 DD DSN=hlq.DCERACF.H,DISP=SHR
//COMPILE.SYSIN DD DSN=hlq.DCERACF.C(SAVEPWS),DISP=SHR
//*
//* COMPILE THE SVPW SERVER STUB
//* ----------------------------
//*
//SVPWSS EXEC DCECC,
// CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN DD DSN=hlq.DCERACF.OBJ(SVPWSS),DISP=SHR
//COMPILE.USERLIB2 DD DSN=hlq.DCERACF.H,DISP=SHR
//COMPILE.SYSIN DD DSN=hlq.DCERACF.C(SVPWSS),DISP=SHR
//*
//* LINK THE SVPW SERVER
//* --------------------
//*
//LINK      EXEC DCELK,OUTFILE='hlq.DCERACF.LOAD'
//USERLIB DD DSN=hlq.DCERACF.OBJ,DISP=SHR
//PLKED.SYSIN DD *
        INCLUDE USERLIB(SVPWSS)
        INCLUDE USERLIB(SAVEPWS)
//LKED.USERLIB DD DSN=hlq.DCERACF.OBJ,DISP=SHR
//LKED.SYSPRINT DD SYSOUT=*
//LKED.SYSIN DD *
        INCLUDE USERLIB(DCERACF)
        NAME    SAVEPWS(R)
//*
```

Figure 59. JCL to Compile and Link, SAVEPWS

B.8.22 SAVEPWSR

```
//ERICFINS JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=64M,
//    MSGLEVEL=(1,1)
//* --------------------------------------------------------------------*
//*                                                                     *
//* Note: CHANGE 'hlq.DCERACF' TO '???.DCERACF' ALL                     *
//*                                                                     *
//* Note: The PARM is the HFS KEYTAB file that has the 'savepw'         *
//*       principal entry.                                              *
//*                                                                     *
//* Note: EUVSJRB5 DD points to the ticket cache to be used.            *
//*                                                                     *
//* --------------------------------------------------------------------*
//*                                                                     *
//* EXECUTE THE SAVEPW SERVER                                           *
//* -------------------------                                           *
//*                                                                     *
//SAVEPWS  EXEC PGM=SAVEPWS,REGION=64M,
//             PARM='//u/ericfin/savepw.key'
//STEPLIB  DD DSN=hlq.DCERACF.LOAD,DISP=SHR
//EUVSKRB5 DD PATH='/u/ericfin/krb5ccname.savepw'
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//*
```

Figure 60. JCL to Run Server, SAVEPWSR

B.8.23 SVPWIDL

```
//ERICFINI JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=8000K,
//    MSGLEVEL=(1,1)
//NEWLK    JCLLIB ORDER=(DCE.SEUVPRC)
//* --------------------------------------------------------------------
//*
//* NOTE: CHANGE 'hlq.DCERACF' TO '???.DCERACF' ALL                   *
//*                                                                   *
//* ------------------------------------------------------------------
//*                                                                   *
//* GENERATE THE STUBS AND IDL HEADER
//* ---------------------------------
//*
//IDLCOMP  EXEC IDL,USERPFX='hlq.DCERACF',DCEPFX='DCE',REGSIZE=8000K,
//    PARMS='SVPW -v -client all -server all -keep all'
//
```

Figure 61. JCL to IDL Compile, SVPWIDL

B.8.24 USERSVC

```
//ERICFIN1 JOB (999,POK),'ERICFIN',NOTIFY=ERICFIN,
//    CLASS=A,MSGCLASS=H,REGION=5000K,
//    MSGLEVEL=(1,1)
//*--------------------------------------------------------------------*
//*
//* NOTE: CHANGE 'hlq.DCERACF' TO '???.DCERACF' ALL                    *
//*                                                                    *
//* NOTE: CHANGE THE USER SVC NUMBER FROM IGC0025E TO YOUR CHOICE      *
//*                                                                    *
//*--------------------------------------------------------------------*
//*                                                                    *
//* ASSEMBLE USERSVC                                                   *
//* ----------------                                                   *
//*                                                                    *
//USERSVC EXEC ASMHC,PARM.C='NODECK,OBJECT'
//C.SYSLIB   DD  DSN=hlq.DCERACF.ASM,DISP=SHR
//           DD  DSN=SYS1.MACLIB,DISP=SHR
//           DD  DSN=SYS1.MODGEN,DISP=SHR
//C.SYSLIN   DD  DSN=hlq.DCERACF.OBJ(USERSVC),DISP=SHR
//C.SYSIN    DD  DSN=hlq.DCERACF.ASM(USERSVC),DISP=SHR
//*
//* COMPILE GETPAC                                                    *
//* --------------
//*
//GETPAC EXEC DCECC,CPARM='SO,LO,NOMAR,NOSEQ'
//COMPILE.SYSLIN   DD DSN=hlq.DCERACF.OBJ(GETPAC),DISP=SHR
//COMPILE.USERLIB2 DD DSN=hlq.DCERACF.H,DISP=SHR
//COMPILE.SYSIN    DD DSN=hlq.DCERACF.C(GETPAC),DISP=SHR
//*
//* LINK USER SVC                                                     *
//* -------------
//*
//LINK  EXEC DCELK,OUTFILE='hlq.DCERACF.LOAD',
//      LPARM='AMODE=31,MAP,RENT,REUS'
//PLKED.USERLIB  DD DSN=hlq.DCERACF.OBJ,DISP=SHR
//PLKED.SYSIN    DD *
 INCLUDE USERLIB(GETPAC)
//LKED.USERLIB   DD DSN=hlq.DCERACF.OBJ,DISP=SHR
//LKED.SYSLMOD   DD DSN=SYS1.LPALIB,DISP=SHR
//LKED.SYSIN     DD *
 INCLUDE USERLIB(USERSVC)
 ENTRY   IGC0025E
 SETCODE AC(1)
 NAME    IGC0025E(R)
//*
```

Figure 62. JCL to Assemble and Link, USERSVC

Appendix C. Structured Programming Macro Reference

The complexity of control flow in a program strongly affects the ease of reading it, the early detection of coding errors, and the effort needed to modify it later. Control flow can usually be simplified (though sometimes at the cost of less efficiency and more redundant code) by restricting the ways in which branches occur. One way to restrict branches is to use only those necessary to implement a few basic structures such as:

- Executing one of two blocks of code according to a true-false condition
- Executing a block of code repeatedly until some limit is reached
- Executing the nth block of code in a given set, where n is previously computed
- Calling another module as a subroutine.

If statements exist for all these structures in a programming language, then they are used exclusively. If some are missing, then simple branches are used to simulate those structures but only in standard patterns. In the case of OS Assembler Language, only the basic branch and branch-and-link instructions are implemented but macros that simulate the first three structures are available.

It has been shown that the four structures above (in fact, the first two) are sufficient to implement any ″proper″ program (that is, with one entry point and one exit) provided that its blocks of code are suitably ordered. It is assumed that the structures may be nested to any depth. The technique of writing programs using only these structures for branching is known as ″structured programming″.

The standard structured programming figures have been implemented for the assembler language programmer through the following three sets of related macros.

- The IF macro set:

  IF
  ELSE (optional)
  ENDIF.

- The DO macro set:

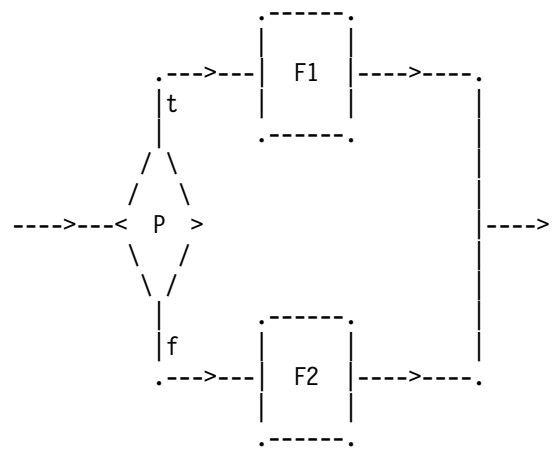
  DO
  ENDDO.

- The CASE macro set:

  CASENTRY
  CASE (one must be present)
  ENDCASE.

C.1 The IF Macro Set

The IF macro set is used to implement the IFTHENELSE program figure. The flowchart for this figure is:



In this figure, the test of the predicate p is represented by the IF macro, which determines whether process F1 or F2 is to be executed. The collector node is represented by the terminator ENDIF macro. The general IF macro set is written:

```
IF  p  THEN

 Code for F1

ELSE

  Code for F2

ENDIF
```

If the ELSE is not used, the flowchart is reduced to one that does not contain function F2 and is written:

```
IF p THEN

  Code for F1

ENDIF
```

The format of the predicate p may take one of the forms discussed below. In each form the keywords AND, OR, ANDIF, and ORIF are optional. THEN is a comment and must be preceded by one or more blanks if used.

PREDICATE		CONNECTOR/ TERMINATOR
IF	condition instruction,parm1,parm2,condition compare-instruction,parm1,condition,parm2	,AND, ,OR, ,ANDIF, ,ORIF, ,THEN,

In the IF examples that follow the parentheses surrounding the predicate are optional.

C.1.1 IF Macro Option A

```
IF   (condition)
```

Option A is used to test the previously set condition code. It uses the Extended mnemonics for the branch instruction or the numeric condition code masks to indicate the condition. The mnemonics allowed are shown in the table below.

```
IF  (H)  THEN

    Code for F1

ELSE

    Code for F2

ENDIF
```

The macros would generate:

```
    BC  15-2,L1

      Code for F1

    BC  15,L2
L1  EQU  *

      Code for F2

L2  EQU  *
```

The same example using a numeric condition code mask would be:

```
IF  (2)  THEN

  Code for F1

ELSE

  Code for F2

ENDIF
```

The same code would be generated.

Case	Mnemonics	Meaning	Complement
After compare instructions	H, GT L, LT E, EQ	high, greater than low, less then equal	NH, LE NL, GE NE
After arithmetic instructions	P M Z O	plus minus zero overflow	NP NM NZ NO
After test under mask instructions	O M Z	ones mixed zeros	NO NM NZ

C.1.2 IF Macro Option B

```
IF   (Instruction mnemonic,Parm1,Parm2,condition)
```

In option B, all four parameters are required. The ″instruction mnemonic″ is any other than a compare, that sets the condition code. (If the condition code has been set previously then option A should be used.) The parameters parm1 and parm2 are the two fields associated with the instruction. ″Condition″ is the value that the condition code mask must assume for the THEN clause to be executed. The ″condition″ parameter is either one of the extended mnemonics given in the above table or a numeric condition code.

An example of option B is:

```
        IF  (TM,BYTE,X'80',Z)  THEN

          Code for F1

        ELSE

          Code for B2

        ENDIF
```

The code generated by the macros for this example would be as follows:

```
        TM  BYTE,X'80'
        BC  15-8,L1

          Code for F1

        BC  15,L2
   L1   EQU  *
```

```
          Code for F2

    L2    EQU  *
```

Option B also provides for three-operand instructions such as those that are available on the System/370. An example is:

```
         IF    (ICM,R1,M3,B2(D2),4)
```

This macro would generate:

```
         ICM  R1,M3,B2(D2)
         BC   15-4,L1
```

C.1.3 IF Macro Option C

```
      IF    (Compare instruction,Parm1,Condition,Parm2)
```

In option C, all four parameters are required. Any compare instruction is valid. However, with a compare instruction, the conditional relation mnemonic appears between parm1 and parm2, instead of after both of them as in option B. In all cases, parm1 and parm2 must agree with the way the user would write them, if he were generating the instruction in assembler language. The condition parameter is either an extended mnemonic from the above table or a mnemonic condition code.

An example of option C is:

```
         IF  (CLI,0(2),EQ,X'40')  THEN

           Code for F1

         ELSE

           Code for F2

         ENDIF
```

The code generated by the macros for this example would be as follows:

```
         CLI  0(2),X'40'
         BC   15-8,L1

           Code for F1

         BC   15,L2
      L1 EQU  *
```

```
      Code for F2

L2 EQU  *
```

Option C also provides for the three-operand CLM compare instruction available on the System/370. An example is:

```
IF  (CLM,R1,M3,NE,B2(D2))
```

C.1.4 IF Macros with Boolean Operators

All the options described in the preceding sections can be combined into longer logical expressions using boolean operators AND, OR, ANDIF, and ORIF. (These are reserved keywords and cannot be used as operands of instructions.) A NOT operator has not been implemented since a complement exists for each of the alphabetic mnemonic condition code indicators described previously.

All logical expressions are scanned from left to right. When the AND and OR connectors are used, the code generated is such that as soon as the expression can be verified as either true or false the appropriate branch to process either the code for F1 or the code for F2 is taken without executing the remaining tests. Statements that are continued onto more than one line must have an X in column 72 of all xmp except the last. Continued xmp must start in column 16.

An example is:

```
IF  (10),OR,
    (AR,R2,R3,NZ),AND,
    (ICM,R1,M3,B2(D2),4)  THEN

  Code for F1

ELSE

  Code for F2

ENDIF
```

The code generated by the macros for this example would be:

```
         BC  10,L1
         AR  R2,R3
         BC  15-7,L2
         ICM  R1,M3,B2(D2)
         BC  15-4,L2
      L1 EQU  *

           Code for F1

         BC  15,L3
      L2 EQU  *

           Code for F2

      L3 EQU  *
```

Note that if the condition code mask setting is 10, upon entering the IF code, the program immediately branches to the F1 code. If it is not 10, and if the next condition code setting is such that the desired relation is not true, the branch is made around the third test to the F2 code. This is done since the AND condition cannot be met if the second relation is false.

The ANDIF and ORIF are used to give a parenthetical grouping capability to the logical expressions. The use of either of these two as connectors of logical groupings, the use of AND or OR indicates a closing parenthesis on the preceding group and an opening parenthesis on the one following. Thus, if the previous example is modified by replacing the AND by an ANDIF, this means that either the first or second condition must be true as well as the third one in order to execute F1.

This example is written as:

```
         IF (10),OR,
            (AR,R2,R3,NZ),ANDIF,
            (ICM,R1,M3,B2(D2),4)  THEN

           Code for F1

         ELSE

           Code for F2

         ENDIF
```

The following code is generated by the macros for this example:

```
         BC  10,L1
         AR  R2,R3
         BC  15-7,L3
      L1 EQU  *
         ICM  R1,M3,B2(D2)
         BC  15-4,L3
      L2 EQU  *
```

```
     Code for F1

   BC  15,L4
L3 EQU  *

     Code for F2

L4 EQU  *
```

In order to better illustrate the effect of the ANDIF and ORIF usage the examples which follow use capital letters to indicate the conditions that are tested.

If the following is written: A OR B AND C, the implied grouping is A OR (B AND C).

With the following: A OR B ANDIF C, the grouping is (A OR B) AND C.

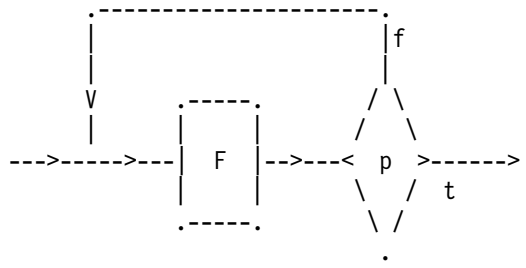
Thus the ORIF may be similarly used: A AND B ORIF C OR D is interpreted as (A AND B) OR (C OR D).

C.2 The DO Macro Set

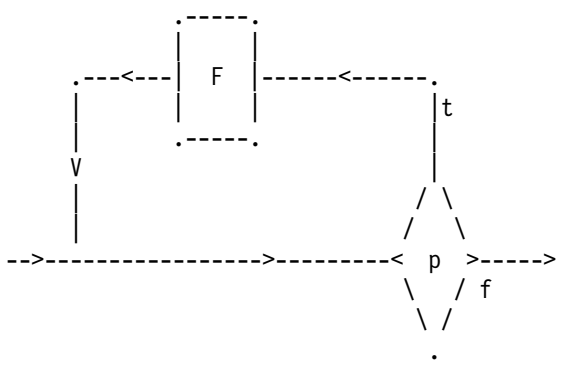
The general DO macro set is written as:

```
DO  P

 Code for F

ENDDO
```

The flowchart represented by this set depends on the keywords used with the predicate p. If the UNTIL or the indexing group of key words (FROM, TO, BY) is used, the flowchart is:



If the WHILE keyword is specified, the flowchart is:



The DO macro is designed with one positional parameter and five possible keywords. The positional parameter may be INF, BXH, or BXLE. The keywords are FROM, TO, BY, WHILE, and UNTIL. The FROM, TO, and BY keywords form an indexing group that is used to specify ranges and increments when indexing through a loop. They indicate loop termination tests, which are made after execution of the function code, and the determination of whether to repeat the loop is made by one of the four indexing instructions: BXH, BXLE, BCT, or BCTR. If an indexing instruction is not given explicitly as a positional parameter, one is derived from the other values given. Infinite looping is also permitted through use of the INF positional parameter.

The function of the UNTIL keyword is similar to that of the loop terminator, except that the determination of whether to repeat one function depends upon the result of any condition code setting instruction. It may not be used with the indexing group.

The WHILE keyword, on the other hand, generates a test prior to entering the function code of the loop. It may be used with either the indexing group or the UNTIL keyword to provide tests at both initiation and termination of the function code.

In all cases, the figure must be terminated by the ENDDO macro.

C.2.1 The DO Indexing Group

The indexing group permits five types of counting and testing to be performed. Each different requirement for counting and testing has a corresponding set of keywords and values, and results in the generation of appropriate loop initialization and termination instructions. The five variations are described in the following paragraphs and are summarized in the table below. The tests to determine which variation is to be used are performed in the order described herein.

In the indexing group, each of the three keywords is permitted to indicate a register designation and an optional value. Thus, an indexing DO statement could appear as:

```
DO  FROM=(Rx,i),TO=(Ry+1,j),BY=(Ry,k)
```

if all keywords in the group were used.

The format of the keywords is keyed to the indexing instructions, in particular BXH and BXLE, and the restrictions on the use of these instructions are carried over into the macros. Therefore, if the BY register Ry is an even number register, then the TO register must be Ry+1. If the BY register Ry is an odd-numbered register, then the TO register Ry+1 must be the same register, and hence the TO and BY values (j and k, respectively) must be identical.

C.2.2 DO Loop Terminator Generation

Case	Type	Keywords	Other Conditions	Result
1	Infinite Loop	Neither FROM WHILE nor UNTIL	INF parameter	BC 15,
2	Explicit Specificat'n	FROM, plus TO and/or BY	BXH parameter	BXH
			BXLE parameter	BXL
3	Counting	FROM (only)	Two values	BCT
			Three values	BCTR
4	Backward Indexing	FROM, TO and BY	FROM and TO numeric, FROM value greater than TO value	BXH
		FROM BY	BY numeric and less than zero	
5	Forward Indexing	Any combination not covered in Cases 1 through 4		BXLE

C.2.3 Infinite Loop

If the programmer wishes to execute a loop until some external event takes place (for example, an end of file), then he may do so by specifying the INF positional parameter.

Thus, coding:

```
DO  INF

  Code for F

ENDDO
```

will result in the generation of:

```
L1 EQU  *

  Code for F

BC  15,L1
```

In order to generate an infinite loop, no FROM, WHILE, or UNTIL keywords can be present. TO and BY keywords, if present, will be ignored.

C.2.4 Explicit Specification

If the programmer wants to specify an explicit BXH or BXLE loop terminator, he may do so by including it in the form of a positional parameter, as follows:

```
DO  BXH,FROM=(Rx,i)TO=(Ry+1,j),BY=(Ry,k)

  Code for F

ENDDO
```

The code generated is:

```
   LA  Rx,i
   LA  Ry+1,j
   LA  Ry,k
L1 EQU  *

     Code for F

   BXH  Rx,Ry,L1
```

The FROM and either the BY or TO keywords must be present in order to provide register designations required for the generation of the BXH/BXLE instruction. The register specified for the BY keyword is used unless it is not present, in which case the one for the TO keyword is used.

C.2.5 Counting

This case applies when a count is to be decremented by one each time, and the loop is to be terminated when the count reaches zero. This is achieved by specifying just the FROM keyword. In the situation where only two parameters are used, a BCT loop terminator is generated.

An example of this is:

```
         DO  FROM=(Rx, number)

           Code for F

         ENDDO
```

In this example the macros would generate the following code:

```
            LA  Rx,number
      L1 EQU  *

           Code for F

         BCT  Rx,L1
```

In order to provide a slightly tighter loop, the FROM keyword may be written with three parameters to designate an additional register. In this case, a BCTR is generated as the loop terminator.

An example of this is as follows:

```
         DO  FROM=(Rx,=A(TEST),Ry)

           Code for F

         ENDDO
```

The macros will generate:

```
            LA  Ry,L1
            L   Rx,=A(TEST)
      L1 EQU  *

           Code for F

         BCTR  Rx,Ry
```

If no value appears in the FROM keyword, the load instruction is not generated.

C.2.6 Backward Indexing

If it is desired to index backward through an array (from high to low storage addresses), then a BXH test is required so that the loop will terminate when the lowest address has been reached. This may be achieved in two ways.

The more explicit of the two ways is specified by use of all three keywords with numeric values for the FROM and TO values i and j, where the FROM value i is greater than the TO value j. Although no test on the BY value k is performed, it should be negative. Also, while the FROM and TO values i and j need not be positive, they are assumed to be negative numeric if and only if a loading minus sign occurs.

Thus, i is greater than j in the following case:

```
         DO  FROM=(Rx,6).TO=(Ry+1,-6),BY=(Ry,-4)

           Code for F

         ENDDO
```

and results in the following code being generated:

```
         LA  Rx,6
         LH  Ry+1,=H'-6'
         LH  Ry,=H'-4'
      L1 EQU  *

           Code for F

         BXH  Rx,Ry,L1
```

If the TO keyword is omitted, and the BY value k is negative numeric (again determined by the presence of a loading minus sign), then it is similarly assumed that backward indexing is desired. Although no test on the register number Ry is performed, it must be odd.

For example: since k is negative, then:

```
         DO  FROM=(Rx,=A(END-START)),BY=(Ry,-2)

           Code for F

         ENDDO
```

will result in the following:

```
         LH  Ry,=H'-2'
      L1 EQU  *

           Code for F

         BXH  Rx,Ry,L1
```

C.2.7 Forward Indexing

If it is desired to index forward through an array (from low to high core addresses), then a BXLE test is required so that the loop will terminate when the highest address has been reached. If no explicit terminator is specified, and if none of the preceding combinations of keywords and values exist, then this type of indexing is assumed, and a BXLE terminator is generated.

An example of this case is:

```
DO  FROM=(Rx,1),TO=(Ry+1,10),BY=(Ry,2)

  Code for F

ENDDO
```

The code generated by the macros would be as follows:

```
LA  Rx,1
LA  Ry+1, 10
LA  Ry,2 L1  EQU  *

  Code for F

BXLE  Rx,Ry,L1
```

C.2.8 Register Initialization

If the user wishes to load a register himself, or the register remains loaded from a previous operation, then omitting the corresponding value field will prevent generation of a register load instruction. If the user supplied one or more of the values i, j, or k, thus indicating that he wishes the macro processor to generate the L, LR, or LA instructions, the following rules apply.

For a positive number greater than zero and less than 4096, an LA is generated. The L or LH instruction will be generated for the case where a value is identified as a negative number (determined by the presence of a leading minus sign), or a positive number greater than 4095. The value will also be converted to a literal, thus generating =F′number′ or =H′number′ as appropriate, and will be substituted as the second operand of the load instruction.

If any value is zero, an SR to clear the designated register will be generated.

In all other cases (nonnumeric or undefined values, as indicated by the type attribute of the macro), an L instruction will be generated. In this case, whatever is present for the value will be directly substituted as the second operand of the instruction. If it is a literal, it is the responsibility of the programmer to supply the equal sign.

The table below summarizes the rules followed in initializing registers.

Value given	Instruction Generated
None	None
Zero	SR Rx,Rx
0 Number 4096	LA Rx,value
Leading minus sign or Number 4096 (Value)	LH Rx,=H'Value'
	L Rx,=F'Value'
	LR Rx, Value
Other	L Rx, Other

C.2.9 The UNTIL/WHILE Keywords

The test generated by the UNTIL keyword, as with those generated by the indexing by the indexing group, is used at the loop termination. That generated by the WHILE keyword, on the other hand, is used to test whether to enter a loop at all prior to its execution. For both keywords, the parameterization is identical to that of the IF macro except that a boolean expression is invalid.

The DOWHILE is coded as follows:

```
DO  WHILE=(TM,FLAGS,X'80',o)

  Code for F

ENDDO
```

The macros in this example would generate the following code:

```
   BC  15,L2
L1 EQU  *

     Code for F

L2 TM  FLAGS,X'80'
   BC  1,L1
```

The DOUNTIL is coded in the same manner as that shown by the following example:

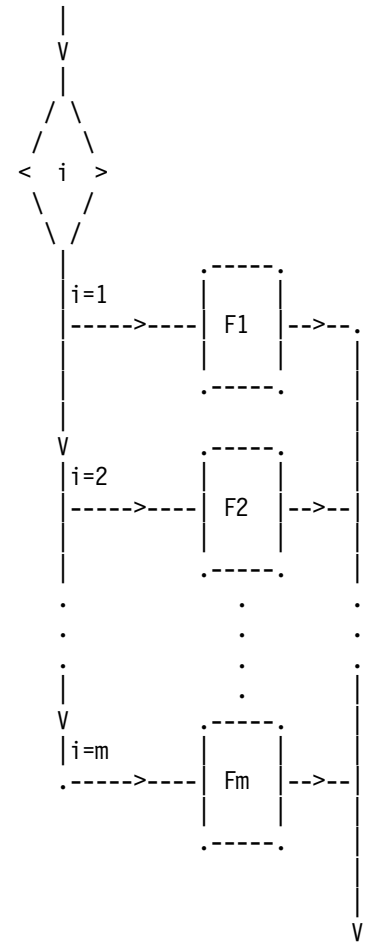
```
DO  UNTIL=(TM,FLAGS,X'80',o)

  Code for F

ENDDO
```

The code generated by the macros used in this example is:

```
L1  EQU  *

      Code for F

    TM  FLAGS,X'80'
    BC  15-1,L1
```

C.3 The CASE Macro Set

The CASE macro set is used to select one of a set of functions for execution, depending on the value of an integer found in a specified register. The determination of which of the functions is to be executed involves the use of either an address vector (sequence of addresses) or a branch vector (sequence of branch instructions). The flowchart for the CASE program figure is:



The macro is written as follows:

```
CASENTRY  Rx,POWER=n,VECTOR=B|BR
  CASE  a, c, d

    Code for F1

  CASE  b, c

    Code for F2

  CASE  f

    Code for F3
    .
    .
    .
    .
    .
  CASE  t

    Code for Fm

ENDCASE
```

Where the case numbers a, b, ..., t are either members of a set of integers greater than zero, or multiples of a power of 2 (for example, 4, 12, and 16). Zero (0) is not a valid case number. Rx is a positional operand that specifies a general register containing the case number. The keyword operands POWER and VECTOR are optional.

The operand POWER=n (n is an integer) refers to a power of 2 and is used to indicate that the case numbers are multiples of that power of 2. Thus, POWER=3 indicates that the case numbers are multiples of 8.

The operand VECTOR=B or VECTOR=BR indicates that a branch vector is to be generated rather than an address vector. Fewer instructions are generated for branch vectors. However, the programmer must be sure that the branch vector table is addressable by the initialization code, that the code for each of the cases is addressable, and that the code after the ENDCASE macro is addressable by a current base register.

Register 0 may not be used as the case value register (Rx).

It is the programmer's responsibility to load the desired case number into Rx and to ensure that it is within the indicated range. The macro expansion then adjusts this value so that the correct CASE is selected. The content of the register indicated in the CASENTRY statement is destroyed and is only required during the execution of the initial code generated by the macro expansion. Hence, it is possible to use the same register for other purposes within the function code for any case(s).

The following is an example of a CASE macro using case numbers of 1, 2, 3, 4 and 5:

```
CASENTRY  Rx
  CASE 2,1,4

    Code for F1

  CASE 5

    Code for F2

ENDCASE
```

This is interpreted to mean that if a 1, 2, or 4 is present in general register Rx, the code for F1 will be executed. If a 5 is present, the code for F2 will be executed. If a value of 3 is in the Rx, no function code is to be executed. In all cases, control is then to be passed to the code after the ENDCASE macro.

The code generated is as follows:

```
         SLA  Rx,2
         A    Rx,L3
         L    Rx,0(rx)
         BCR  15,Rx
      L3 DC   A(l1)
      L4 EQU  *

           Code for F1

         L    Rx,L1
         BCR  15,Rx
      L5 EQU  *

           Code for F2

         L    Rx,L1
         BCR  15,Rx
      L1 DC   A(L2)
         DC   A(L4)
         DC   A(L4)
         DC   A(L2)
         DC   A(L4)
         DC   A(L5)
      L2 EQU  *
```

An example of a CASE macro using a branch vector and case numbers that are multiples of 8 is:

```
         CASENTRY  Rx,POWER-3,VECTOR=B
           CASE 8,24

             Code for F1

           CASE  16,32

             Code for F2

         ENDCASE
```

This code generated is as follows:

```
         SRA  Rx,3-2
         BC   15,L2(Rx)
      L3 EQU  *

           Code for F1

         BC   15,L2
      L4 EQU  *

           Code for F2

         BC   15,L2
      L1 BC   15,L2
```

```
   BC   15,L3
   BC   15,L4
   BC   15,L3
   BC   15,L4
L2 EQU  *
```

Index

R

S

U

ITSO Technical Bulletin Evaluation RED000

International Technical Support Organization
MVS/ESA OpenEdition DCE:
RACF and DCE Security Interoperation
April 1995

Publication No. GG24-2526-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction	____		
Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

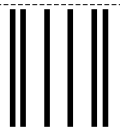
_____ _____
Name Address

_____ _____
Company or Organization

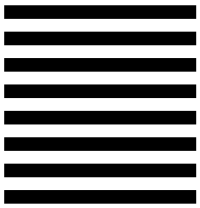
_____ _____
Phone No.

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Mail Station P099
522 SOUTH ROAD
POUGHKEEPSIE NY
USA 12601-5400

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.