z/OS

Integrated Security Services Open Cryptographic Enhanced Plug-ins Application Programming

SC24-5925-01

z/OS

Integrated Security Services Open Cryptographic Enhanced Plug-ins Application Programming

SC24-5925-01

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 37.

Second Edition (September 2007)

This edition applies to Version 1 Release 9 of z/OS (5694-A01) and to all subsequent releases of this product until otherwise indicated in new editions.

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

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States and Canada): 1+845+432-9405
FAX (Other Countries):

Your International Access Code +1+845+432-9405

IBMLink (United States customers only): IBMUSM10(MHVRCFS)
IBM Mail Exchange: USIB6TC9 at IBMMAIL
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

About This Book

This book contains information about Open Cryptographic Enhanced Plug-ins (OCEP), which is a component of z/OS Integrated Security Services. Integrated Security Services works with the following components:

- Resource Access Control Facility (RACF)
- DCE Security Server
- z/OS Firewall Technologies
- Lightweight Directory Access Protocol (LDAP) Server, which includes client and server function
- Open Cryptographic Enhanced Plug-ins

Purpose of This Book

This book describes an overview of OCEP, the service provider modules that it provides, and how those modules work with Open Cryptographic Services Facility (OCSF) and Resource Access Control Facility (RACF) which comprises the Security Server component.

This book describes how to install and register the OCEP service provider modules for use with the OCSF Framework. In addition, it describes the application programming interfaces (APIs) that OCEP supports.

OCSF, which is a derivative of the IBM Keyworks technology, is an implementation of the Common Data Security Architecture (CDSA) for applications that run in the z/OS UNIX System Services (z/OS UNIX) environment.

Who Should Use This Book

This book is written for programmers who have experience with writing and supporting security applications. Knowledge of the OCSF Framework and the components of the OS/390 Security Server is required. In addition, knowledge of the services provided by Integrated Cryptographic Service Facility (ICSF) is also helpful.

This book also provides information to help system programmers configure OCEP for use on their OS/390 systems. It describes how to install and register the OCEP service provider modules with the OCSF Framework.

In addition, this book should be used by application programmers who intend to use the functions and APIs supported by OCEP.

What This Book Contains

This book describes the OCEP service provider modules and how they are intended to be used with the framework provided by OCSF. It also describes how these service provider modules enable applications to use Security Server, or an equivalent product, to provide security functions relating to digital certificates.

Where to Find More Information

For detailed information about the OCSF Framework, refer to the following publications:

- *z/OS Open Cryptographic Services Facility Application Programming*
- *z/OS Open Cryptographic Services Facility Service Provider Module Developer's Guide and Reference*

For information about RACF's support for digital certificates and its interaction with OCEP, refer to the following publications:

- *z/OS Security Server RACF Callable Services*
- *z/OS Security Server RACF Command Language Reference*
- *z/OS Security Server RACF Security Administrator's Guide*

For information about the publications that support the other elements of OS/390, see the *z/OS Information Roadmap*.

Summary of changes

Summary of changes
for SC24-5925-01
z/OS Version 1 Release 9

The document contains information previously presented in SC24-5925-00, which supports z/OS Version 1 Release 1.

New information

These items are new for this release:

- Support referencing IBM Software Cryptographic Service Provider 2 has been added.
- Support referencing IBM Weak Software Cryptographic Service Provider 2 has been added.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Chapter 1. Introducing OCEP

This section introduces the services that are provided by Open Cryptographic Enhanced Plug-ins (OCEP) and their relationship with Open Cryptographic Services Facility (OCSF) and Resource Access Control Facility (RACF). (Any external security manager product that provides equivalent support may also be used.)

Overview

As Figure 1 on page 2 shows, OCEP consists of two service provider modules (which are also called "plug-ins") that are intended to be used with the Open Cryptographic Services Facility (OCSF) Framework:

- Trust Policy
- Data Storage Library

These service provider modules enable applications to use z/OS Security Server (RACF), or equivalent product, to provide security functions for digital certificates and key rings.

The OCEP service provider modules implement a subset of the application programming interfaces (APIs) that are defined by OCSF. Applications can use these OCEP service provider modules, and their supported APIs, to retrieve and use digital certificates and private keys that are stored in the RACF database on a z/OS system.

In addition to the OCSF Framework, the OCEP service provider modules are intended to work with the OCSF Certificate Library and Cryptographic Service Provider modules. As Figure 1 on page 2 shows, the OCSF Framework itself manages the interactions between the service provider modules and the applications that use them.

For a detailed description of the OCSF application programming interfaces and the service provider modules that OCSF supports, see the following publications:

- *z/OS Open Cryptographic Services Facility Application Programming*.
- *z/OS Open Cryptographic Services Facility Service Provider Module Developer's Guide and Reference*

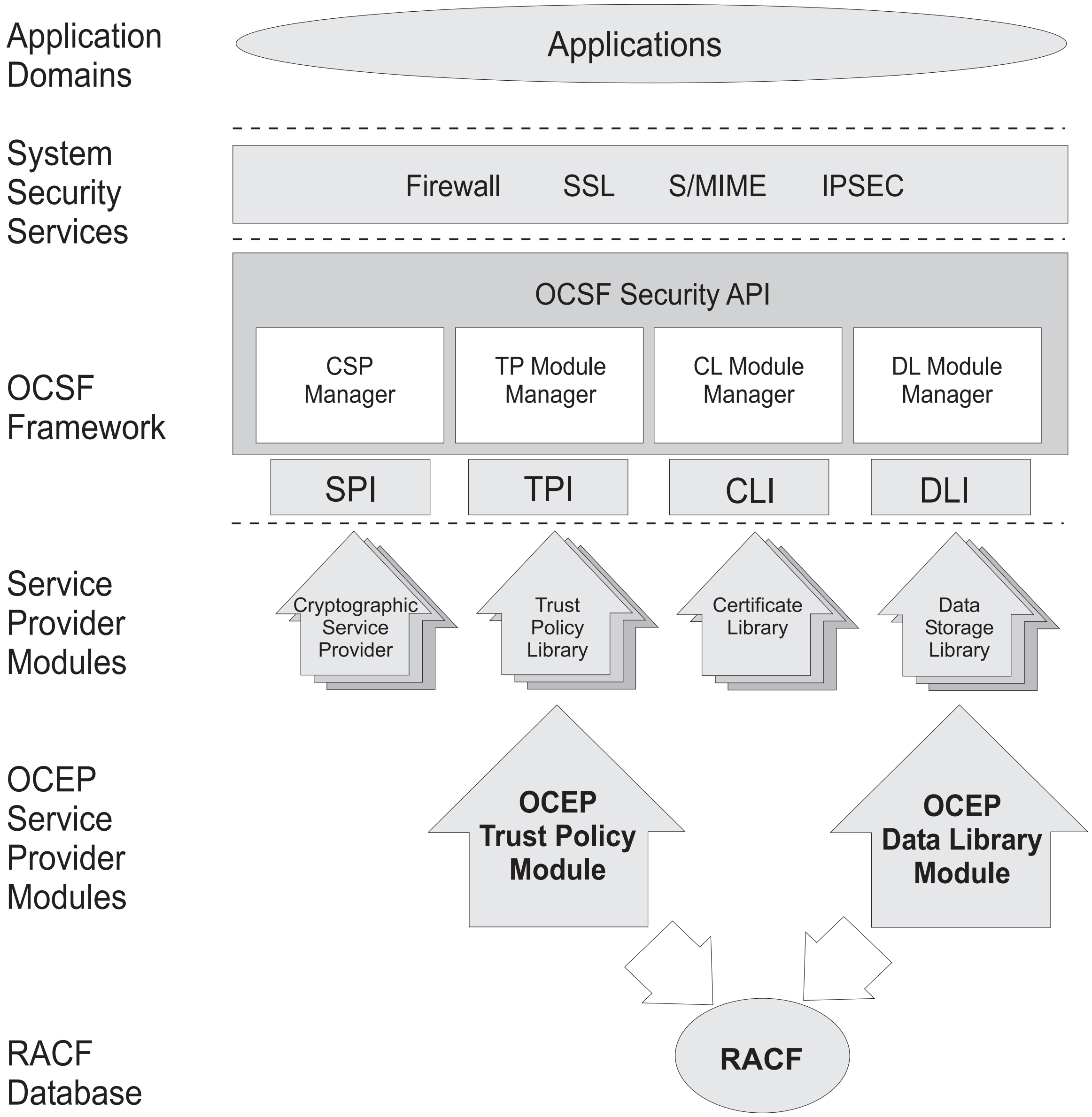

Figure 1. Overview of the OCEP and OCSF Infrastructure

OCEP Trust Policy

In the OCSF Framework, a trust policy (TP) service provider module implements policies that are defined by Certificate Authorities (CAs) and institutions. These policies define the level of trust that is required before certain actions can be performed. When a TP function has determined the trustworthiness of performing an action, the TP function may invoke other functions in a certificate library and a data storage library service provider module to carry out the mechanics of the approved action.

The OCEP Trust Policy service provider module implements the trust policy that is defined by a specific RACF key ring. (The OCEP Trust Policy service provider module, however, does not provide Certificate Revocation List support, as defined by OCSF.) It determines the validity of a certificate group (also called a "chain") by checking if the chain originated from a trusted certificate authority or if the first entity in the chain is connected to the key ring as a SITE certificate. A SITE certificate is one that the RACF administrator has explicitly defined and added as a trusted certificate.

For each digital certificate in the chain, the OCEP Trust Policy service provider module checks the signatures and ensures that the certificate has not been marked as not trusted by RACF. When a certificate is defined, it is marked as being trusted or not trusted by specifying the TRUST or NOTRUST operand, respectively, on the RACDCERT command. When a certificate is trusted, it indicates that the certificate is valid for the user, site, or the issuing certificate authority. It also indicates that the private key to this certificate has not been compromised.

The chain must originate from a certificate authority that is trusted. You do not have to use the RACDCERT command to add each digital certificate in that chain to RACF. However, if an individual certificate has been added to RACF, it must be marked as trusted; if not, the verification will fail and RACF will not use it to map to a user ID.

The OCEP Trust Policy must use the OCEP Data Storage Library as its data library service provider module. In addition, the OCEP Trust Policy uses the IBM Certificate Library, Version 1 as its certificate library service provider. This module, which is provided with OCSF, verifies the syntax of the fields within the specific types of digital certificates. The OCEP Trust Policy also works with one of the cryptographic service providers that is supplied with OCSF. These service provider modules handle the cryptographic functions and policies that are associated with their specific cryptographic algorithms:

- IBM Software Cryptographic Service Provider, Version 1
- IBM Software Cryptographic Service Provider 2, Version 1
- IBM Weak Software Cryptographic Service Provider, Version 1
- IBM Weak Software Cryptographic Service Provider 2, Version 1

For more information about the OCEP Trust Policy service provider module and the supported API, see Chapter 3, “Using Trust Policy Services,” on page 13. For information about the certificate library and cryptographic service provider modules that are provided in OCSF, refer to the *z/OS Open Cryptographic Services Facility Application Programming*.

OCEP Data Storage Library

Within the OCSF framework, a data storage library service provider module provides persistent storage of security-related objects, such as digital certificates and keys. The OCEP Data Storage Library service provider module is designed to give applications read-only access to key ring information that has been defined and stored in the RACF database.

When the proper authorizations are established, OCEP can access this information from the RACF database. As Table 1 on page 4 shows, an application can use the OCEP Data Storage Library service provider module to query specific fields in the certificate record.

Table 1. Queriable Fields in the Certificate Record

Field Name	Description	Length
Label	The value that identifies the certificate; it must be unique within the certificate class and user ID. For example, the label “CA Cert” may be used for a certificate for an individual user and for a certificate authority’s certificate. Also, two different users may mark their private keys as “My Key”.	1-32 characters (specified in RACF)
Subject DN	The DER-encoded X.500 Subject’s Distinguished Name; it is not unique to this certificate.	256 bytes or less
Default for Ring Attribute	A binary boolean field that indicates if a default is specified; the value is unique to this certificate: **Zero** Not default **Nonzero** Default	4 bytes

In response to a query, the following information about the certificate will be returned to the application:

- DER-encoded certificate
- Private key for a user certificate, if it exists and if the calling user ID owns this certificate
- RACF user ID that owns the certificate
- Label associated with this certificate
- Subject DN
- Key type
- Key size

This information is only returned for certificates that have been marked as trusted by RACF. If the certificate is not trusted, it will not be returned to the application.

For more information about the OCEP Data Library service provider module and the supported APIs, see Chapter 4, “Using Data Storage Library Services,” on page 23.

z/OS Security Server (RACF) Support

In addition to supporting profiles for digital certificates, the RACF database supports the following classes of certificates (in the OCSF Framework, this is known as “semantic information”). Users who have the proper authority can issue a series of RACDCERT commands to create the certificate and key pairs and populate the RACF database with this information:

- User (server) certificates with optional private keys stored under the owning user ID
- Certificate Authorities (no private keys) that are stored at the system level under a unique user ID
- Site certificates (no private keys) that are stored at the system level under another unique user ID

In addition, RACF supports the concept of “user-defined key rings” (in the OCSF Framework, these are known as “data stores”). A key ring is stored under the owning user ID and may contain any of the preceding types of certificates. Entries in a key ring point to certificate records and contain additional attributes, such as:

- Default certificate/key
- Ring usage for the certificate/key

  For example, the user key may be marked as a trusted root. The certificate record would still exist at the user level but it would be treated as a certificate authority for this key ring only.
- Private key type

  This may be an Integrated Cryptographic Service Facility (ICSF) key token label or a non-ICSF key
- Private key bit size

For more information about RACF's support of digital certificates, see the *z/OS Security Server RACF Security Administrator's Guide*. For information about the RACDCERT command, refer to the *z/OS Security Server RACF Command Language Reference*.

For more information about ICSF key tokens, refer to the *z/OS Cryptographic Services ICSF Application Programmer's Guide* and the *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Developing Security Applications

The OCEP service provider modules are designed to plug in to the OCSF Framework. As such, applications that wish to use these service provider modules must understand and follow the OCSF requirements and conventions. For example, OCSF provides a set of APIs to perform core services, such as:

- Installing and attaching service provider modules

  The calling application uses the OCSF CSSM_ModuleAttach function, for example, to attach the specified OCEP service provider modules. CSSM_ModuleAttach then returns a handle value that represents a unique pairing between the calling application and the specific OCEP service provider module. The calling routine must then specify this handle when it invokes an API that is supported by an OCEP service provider. See page 21 for an example.
- Querying the OCSF registry of available service provider modules
- Enabling calls to other APIs
- Managing storage
- Managing errors

In addition, because service provider modules may implement the OCSF APIs differently, you should be aware of any differences between the parameters that are supported. For example, OCSF also provides trust policy and data storage library service provider modules. However, the way in which the APIs are implemented by these OCSF service provider modules support differs from the way they are implemented by OCEP. You should review your applications to ensure that they can correctly use the APIs, as they are supported by the OCEP service provider modules.

For more information about these OCSF requirements, refer to the *z/OS Open Cryptographic Services Facility Service Provider Module Developer's Guide and Reference*.

Chapter 2. Configuring and Getting Started

This chapter describes the procedures that you need to perform after you have completed the installation of the Open Cryptographic Enhanced Plug-ins (OCEP) code on your system. For information about these installation procedures, see the *z/OS Planning for Installation* book and the *z/OS Program Directory* that is supplied with your product order.

Verifying the OCSF Installation and Configuration

Before you can run any applications that use the OCEP service provider modules, you must first ensure that several tasks have been completed for Open Cryptographic Services Facility (OCSF). The following items must be reviewed and completed:

- The OCSF code must be properly installed and configured on your system.
- Any necessary security authorizations must be granted and program controls must be established.
- The required RACF FACILITY class profiles (CDS.*) must be defined for OCSF.
- z/OS user identities must be authorized to access the CDS.* FACILITY class profiles.

For more information about the configuration requirements for OCSF, see the *z/OS Open Cryptographic Services Facility Application Programming*.

Configuring the OCEP Installation

The following sections describe the actions that are required to configure OCEP for use on your system.

Authorizing Daemon and User Identities

IBM recommends that you assign unique z/OS and z/OS UNIX user identifiers (UIDs) to the daemons and applications that are authorized to use OCEP and OCSF services. This approach will maintain individual accountability for applications that are accessing cryptographic services on z/OS.

For example, assume that the following daemon application needs to use OCEP and OCSF services on z/OS. This daemon runs under the z/OS shell and the application is started by the daemon's profile.

UID	RACF Identity (User ID)	Home Directory
25	G092799	`/u/apps/g092799`

To create a RACF user profile with an OMVS segment, you would issue the following RACF ADDUSER command:

```
adduser g092799 omvs(uid(25) home('/u/apps/g092799') program('/bin/sh'))
```

For more information about how to define a RACF user ID, see the *z/OS Security Server RACF Command Language Reference* and the *z/OS Security Server RACF Security Administrator's Guide*.

In addition, IBM recommends that the OCEP installation and verification scripts (see page 10 and page 11) are run from a superuser; that is, a user ID that has been defined with a UID of 0.

For more information about how to define entities for daemons and applications on z/OS, see the *z/OS UNIX System Services Planning* book.

Granting Access to RACF FACILITY Class Profiles

To use the services offered by OCEP, the user IDs that are associated with the daemon applications must be authorized to access RACF FACILITY class profiles. See Table 2 for a list of these FACILITY class profiles and the type of access that is required.

Table 2. Required FACILITY Class Profiles

FACILITY Class Profile	Access	Explanation
IRR.DIGTCERT.LIST	READ	Enables the caller to use the CSSM_TP_CertGroupVerify function.
IRR.DIGTCERT.LISTRING	READ	Enables the caller to use the CSSM_DL_DataGetFirst and the CSSM_TP_CertGroupVerify functions to retrieve the contents of a key ring that is associated with the user's own user ID.
	UPDATE	Enables the caller to use the CSSM_DL_DataGetFirst and the CSSM_TP_CertGroupVerify functions to retrieve the contents of a key ring that is associated with another user's user ID.

In addition, these user IDs must be authorized to access the CDS.* FACILITY class profiles that are required to access the OCSF Framework.

To define these FACILITY class profiles, you would issue the following RDEFINE commands:

```
rdefine facility irr.digtcert.list uacc (none)
rdefine facility irr.digtcert.listring uacc (none)
```

Next, the user ID that is associated with the daemon or application that will call OCEP must be authorized to use the new FACILITY class profiles. For example, to permit the user ID G092799 to access these class profiles, you would issue the following RACF PERMIT commands:

```
permit irr.digtcert.list class(facility) id(g092799) acc(read)
permit irr.digtcert.listring class(facility) id(g092799) acc(read)
```

Depending on the specific requirements of the application, you may also need to authorize the daemon user ID to access other class profiles.

For easier administration, you can also define a group for the user IDs that are associated with the applications that will use OCEP. This group can then be permitted to access the appropriate RACF FACILITY class profiles. Individual users can then be connected, as needed, to the group.

For more information about how to define RACF groups and grant access to the FACILITY class profiles, see the *z/OS Security Server RACF Command Language Reference* and the *z/OS Security Server RACF Security Administrator's Guide*.

For more information about the class authorizations that are required for OCSF, refer to the *z/OS Open Cryptographic Services Facility Application Programming*.

Establishing Program Control

Program control is the concept of having "trusted" applications. Your installation can define libraries to RACF where these trusted applications will reside. You can activate program control on your system by issuing the RACF command SETROPTS WHEN(PROGRAM). When program control is active, processes will be marked "dirty" if they attempt to load programs from libraries that are not trusted.

z/OS UNIX also has the concept of trusted applications. In the UNIX file system, executable files may be tagged with the program-controlled extended attribute. If a user issues an z/OS shell command or runs a program that does not have the program-controlled extended attribute, the process becomes "dirty"; in either case, the process is never "cleaned". That is, the "dirty bit" remains on, which will cause certain services to fail as a result.

Establishing Program Control in RACF

By protecting load modules, your installation can establish controls over who can run certain programs and can, in turn, treat those programs as assets. You can protect individual load modules (programs) by creating a profile for the program in the RACF PROGRAM general resource class. A program that is protected by a profile in the PROGRAM class is called a controlled program. When RACF program control is activated on your system, OCEP also requires the following program libraries to be program-controlled:

- Language Environment, which includes the C/C++ run-time libraries
- Integrated Cryptographic Service Facility (ICSF), if it is used

For more information about RACF program control, refer to the *z/OS Security Server RACF Security Administrator's Guide*.

Establishing Program Control in HFS

You can mark programs and dynamically loaded libraries (DLLs) in the hierarchical file system (HFS) as being controlled ("trusted"). To do so, you must turn on the program-controlled extended attribute for the HFS file that contains the program or DLL. To turn on this extended attribute, issue the following z/OS UNIX shell command:

```
extattr +p filename
```

The OCSF dynamic link libraries and the files that comprise the OCEP service provider modules must have the program-controlled extended attribute. To check if a file has the program-controlled extended attribute, issue the z/OS shell command **ls** with the **-E** option. In the following example, this command is issued to verify that the program-controlled attribute is set for an OCEP file called ibmoceptp.so:

```
$ cd /usr/lpp/ocsf/addins
$ ls -E ibmoceptp.so
-rwxr-xr-x -ps 2 ROOT   SYS1    737280 Nov 3 22:07 ibmoceptp.so
```

The **p** flag in the command output indicates that this file has the program-controlled extended attribute. See the *z/OS UNIX System Services Command*

Reference and the *z/OS UNIX System Services Planning* book for more information about these z/OS shell commands and the program-controlled attribute.

Refreshing RACF Data

After all of the OS/390 Security Server (RACF) definitions have been made, the FACILITY class must be refreshed if it is RACLISTed. To do so, issue the following command:

```
setropts raclist(facility) refresh
```

If the FACILITY class is not active, you may activate it with the following command:

```
setropts classact(facility)
```

If you added members to the PROGRAM class profiles, program control for those members will not be in effect until you issue the following command:

```
setropts when(program) refresh
```

For more information about refreshing RACF data, see the *z/OS Security Server RACF Security Administrator's Guide*. For complete command syntax information, refer to the *z/OS Security Server RACF Command Language Reference*.

Installing the OCEP Code

OCEP provides an installation script, called **ocep_install**, that installs the OCEP code and registers the service provider modules with the OCSF Framework. You must run the OCEP installation script from an z/OS shell session. IBM recommends that the script be run from a superuser, which is a user ID that has been defined with a UID of 0.

To install the OCEP service provider modules, perform the following steps:

1. Ensure that OCSF has been properly installed on your system by running the install verification procedure (IVP), **ocsf_baseivp**.

   For more information about installing OCSF and running the verification procedure, refer to the *z/OS Open Cryptographic Services Facility Application Programming*.
2. Go to the directory where OCEP is installed, for example:
   ```
   cd /usr/lpp/ocsf/bin
   ```
3. Run the OCEP installation script:
   ```
   ocep_install
   ```

   You will receive the following output:
   ```
   Installing IBMOCEPTP...
   Addin successfully installed.
   Installing IBMOCEPDL...
   Addin successfully installed.
   ```

Refer to the README.ocep_ivp file in the `/user/lpp/ocsf/ivp` directory for more information about this installation script.

Verifying OCEP Installation

After you have completed the steps described on page 10, you must run **ocep_ivp**, the OCEP installation verification program, to ensure that the OCEP code is installed and configured correctly. As with the installation script, IBM recommends that this IVP be run from a superuser, which is a user ID that has been defined with a UID of 0.

To do so, perform the following steps:

1. Go to the directory that contains the IVP, for example:
   ```
   cd /usr/lpp/ocsf/ivp
   ```
2. Run the OCEP IVP program:
   ```
   ocep_ivp
   ```
   You will receive the following output:
   ```
   Starting OCEP IVP

   Initializing CSSM
   CSSM Initialized

   Attaching ibmocepdl
   Attach successful, Detaching ibmocepdl
   Detach of ibmocepdl successful

   Attaching ibmoceptp
   Attach successful, Detaching ibmoceptp
   Detach of ibmoceptp successful

   Completed OCEP IVP
   ```

For more information about the installation verification procedure, see the README.ocep_ivp file in the `/user/lpp/ocsf/ivp` directory.

Uninstalling the OCEP Code

If you do not want to make OCEP available for use by applications, you can run the **ocep_uninstall** script, which is provided with OCEP. When you invoke this script, the OCEP service provider modules will no longer be registered to the OCSF Framework. IBM recommends that this script be run from a superuser.

To run this script, perform the following steps:

1. Go to the directory where OCEP is installed, for example:
   ```
   cd /usr/lpp/ocsf/bin
   ```
2. Run the OCEP script:
   ```
   ocep_uninstall
   ```
   You will receive the following output:
   ```
   Uninstalling IBMOCEPTP...
   Addin successfully uninstalled.
   Uninstalling IBMOCEPDL...
   Addin successfully uninstalled.
   ```

Chapter 3. Using Trust Policy Services

This section describes the OCEP Trust Policy service provider module. It also describes its implementation of the trust policy API, as defined in the OCSF Framework.

Using the Trust Policy Module

After you complete the installation steps described in “Installing the OCEP Code” on page 10, the following OCEP Trust Policy service provider files are available for use on your system.

Function	Name	Directory Location
Header File	ibmoceptp.h	`/user/lpp/ocsf/include`
Executable Module	ibmoceptp.so	`/user/lpp/ocsf/addins`

To use the OCEP Trust Policy, an application must explicitly attach this service provider module. To do so, the application must use CSSM_ModuleAttach, which is an API provided by OCSF, to attach the specific GUID for the module. In turn, CSSM_ModuleAttach returns a handle that uniquely represents the pairing of the service provider module and the calling application.

The following GUID identifies the OCEP Trust Policy module; this GUID and other related constants are defined in the ibmoceptp.h header file:

```
// {5E43B291-1C38-11d2-8688-0004ACF320BC}
static const CSSM_GUID IBMOCEPTP_GUID =
{ 0x5e43b291, 0x1c38, 0x11d2, { 0x86, 0x88, 0x0, 0x4, 0xac, 0xf3, 0x20, 0xbc } };
```

For more information about the CSSM_ModuleAttach function and developing security applications, see the *z/OS Open Cryptographic Services Facility Application Programming*.

Supported Trust Policy API

Table 3 summarizes the trust policy functions that are defined in the OCSF Framework and how they are supported by the OCEP Trust Policy service provider module.

Table 3. Trust Policy Library Functions that are Supported by OCEP

Function Name	Supported	Comments
CSSM_TP_ApplyCrlToDb	No	
CSSM_TP_CertGroupConstruct	No	
CSSM_TP_CertGroupPrune	No	
CSSM_TP_CertGroupVerify	Yes	See page 15.
CSSM_TP_CertRevoke	No	
CSSM_TP_CertSign	No	
CSSM_TP_CrlSign	No	
CSSM_TP_CrlVerify	No	

Table 3. Trust Policy Library Functions that are Supported by OCEP (continued)

Function Name	Supported	Comments
CSSM_TP_PassThrough	No	

Note: The following section provides an overview of the API that is supported by the OCEP Trust Policy service provider module. Only the parameters and values that are unique to OCEP's implementation are described.

For a full description of the syntax and supporting parameters of the remaining APIs that are implemented in the OCSF Framework, refer to the *z/OS Open Cryptographic Services Facility Application Programming*.

CSSM_TP_CertGroupVerify

Description

This function verifies a certificate chain, based on the Certificate Authorities and SITE certificates that are contained within the key ring.

Format

```
CSSM_BOOL CSSMAPI CSSM_TP_CertGroupVerify
        (CSSM_TP_HANDLE TPHandle,
         CSSM_CL_HANDLE CLHandle,
         CSSM_DL_DB_LIST_PTR DBList,
         CSSM_CSP_HANDLE CSPHandle,
         const CSSM_FIELD_PTR PolicyIdentifiers,
         uint32 NumberofPolicyIdentifiers,
         CSSM_TP_STOP_ON VerificationAbortOn,
         const CSSM_CERTGROUP_PTR CertToBeVerified,
         const CSSM_DATA_PTR AnchorCerts,
         uint32 NumberofAnchorCerts,
         const CSSM_FIELD_PTR VerifyScope,
         uint32 ScopeSize,
         CSSM_TP_ACTION Action,
         const CSSM_DATA_PTR Data,
         CSSM_DATA_PTR *Evidence,
         uint32 *EvidenceSize)
```

Parameters

TPHandle **(input)**
the handle for this trust policy service provider module.

CLHandle **(input)**
specifies the handle to the required certificate library service provider module, IBM Certificate Library, Version 1. This service provider module is provided in OCSF and it must be attached by the calling application.

DBList **(input)**
identifies one DL and DB handle pair that represents a RACF key ring that was previously opened by a call to CSSM_DL_DbOpen. The *DLHandle* must be the handle that was returned by CSSM_ModuleAttach when the OCEP Data Storage Library service provider module was attached. This *DLHandle* is also specified on calls to the CSSM_DL_DbOpen API.

CSPHandle **(input)**
specifies the handle of one of the following cryptographic service provider modules. These service provider modules are provided in OCSF; the selected service provider module must also be attached by the calling application:

- IBM Software Cryptographic Service Provider, Version 1
- IBM Software Cryptographic Service Provider 2, Version 1
- IBM Weak Software Cryptographic Service Provider, Version 1
- IBM Weak Software Cryptographic Service Provider 2, Version 1

PolicyIdentifiers **(input)**
this parameter is ignored and may be specified as NULL.

NumberofPolicyIdentifiers **(input)**
this parameter is ignored and may be specified as 0.

VerificationAbortOn **(input)**
this parameter is ignored and may be specified as CSSM_TP_STOP_ON_POLICY.

CertToBeVerified **(input)**
a pointer to the CSSM_CERTGROUP structure containing a certificate that has at least one signed certificate for verification. An unsigned certificate template cannot be verified.

AnchorCerts **(input)**
this parameter is ignored and may be specified as NULL.

NumberofAnchorCerts **(input)**
this parameter is ignored and may be specified as 0.

VerifyScope **(input)**
this parameter is ignored and may be specified as NULL.

ScopeSize **(input)**
this parameter is ignored and may be specified as 0.

Action **(input)**
this parameter is ignored and may be specified as 0.

Data **(input)**
this parameter is ignored and may be specified as NULL.

Evidence **(input)**
this parameter is ignored and may be specified as NULL.

EvidenceSize **(output)**
this parameter is ignored and may be specified as 0.

Error Codes

Table 4 lists the error codes that are unique to OCEP’s support of the Trust Policy service provider module.

Table 4. OCEP Trust Policy Error Codes

Decimal Value	Error Description
8010	CA certificate not found in key ring
8011	Certificate chain not trusted

For information about the OCSF APIs that perform error reporting and recovery, plus a list of other OCSF-defined error codes, refer to the *z/OS Open Cryptographic Services Facility Application Programming*.

Trust Policy Example

Figure 2 shows a sample program that uses the trust policy API supported by OCEP. The **highlighted** entries demonstrate how to attach this service provider module and invoke the API; this sample also includes statements to attach the OCEP Data Storage Library service provider module.

```
/**********************************************************************
 *  File name: oceptptest.c                                           *
 *  Description: Sample program to execute TP_CertGroupVerify         *
 **********************************************************************/

/* required header files */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <cssm.h>
#include <cssmapi.h>
#include <cssmtype.h>
#include <ibmcl.h>
#include <ibmswcsp.h>
#include <ibmocepdl.h>
#include <ibmoceptp.h>
#include <unistd.h>

/* function prototypes */
CSSM_RETURN  errorMsg(char * );
CSSM_RETURN  buildCertGroup(CSSM_CERTGROUP *, char * [], uint32);
void         freeCertGroup(CSSM_CERTGROUP *);
CSSM_RETURN  openDB(char *);
void         closeDB(void);
CSSM_RETURN  attachPlugins(void);
void         detachPlugins(void);

/* global pointers */
CSSM_CL_HANDLE       ibm_cl_handle;
CSSM_CSP_HANDLE      ibm_csp_handle;
CSSM_TP_HANDLE       ibm_tp_handle;
CSSM_DL_HANDLE       ibm_dl_handle;
CSSM_DL_DB_HANDLE    dl_db_handle;
CSSM_DL_DB_LIST      ibm_dl_db_list;
CSSM_API_MEMORY_FUNCS MemoryFuncs;
```

Figure 2. Example Code Using the OCEP Trust Policy APIs (Part 1 of 5)

```
/* memory functions */
void myfree ( void *MemPtr, void *AllocRef )
{ free (MemPtr); }

void *mymalloc ( unsigned int Size, void *AllocRef )
{ return malloc (Size); }

void *myrealloc (void *MemPtr, unsigned int Size, void *AllocRef)
{ return realloc (MemPtr, Size); }

void *mycalloc (unsigned int Num, unsigned int Size, void *AllocRef)
{ return calloc (Num, Size); }

/*********************************************************************
 *  name: main                                                       *
 *********************************************************************/
 void main(int argc, char *argv[])
 {
   CSSM_VERSION cssm_version = {CSSM_MAJOR, CSSM_MINOR};
   CSSM_CERTGROUP   certGroup;
   char           * ringName = argv[1];

   if (argc < 3)
   {
     printf("Too few parameters specified.\n");
     printf("Usage: oceptptest userid/keyring certfile1 certfile2 ...\n");
     return;
   }

   MemoryFuncs.malloc_func  = mymalloc;
   MemoryFuncs.free_func    = myfree;
   MemoryFuncs.realloc_func = myrealloc;
   MemoryFuncs.calloc_func  = mycalloc;

   if (buildCertGroup(&certGroup, &argv[2], argc-2) != CSSM_OK) return;

   if (CSSM_Init(&cssm_version, &MemoryFuncs, NULL) != CSSM_OK)
   {
      errorMsg("Failed CSSM_Init");
      return;
   }
   if ((attachPlugins() == CSSM_OK) &&
       (openDB(ringName) == CSSM_OK))
   {
      if (CSSM_TP_CertGroupVerify (
               ibm_tp_handle,
               ibm_cl_handle,
               &ibm_dl_db_list,
               ibm_csp_handle,
               NULL, 0, CSSM_TP_STOP_ON_POLICY,
               &certGroup,
               NULL, 0, NULL, 0, 0, NULL, NULL, 0))

             printf("Certificate verification succeeded\n");
      else errorMsg("Certificate verification failed");
   }
   closeDB();
   detachPlugins();
   freeCertGroup(&certGroup);

   return;
 }
```

Figure 2. Example Code Using the OCEP Trust Policy APIs (Part 2 of 5)

```
/*********************************************************************
 *  name: errorMsg - Show error message and error code               *
 *********************************************************************/

 CSSM_RETURN errorMsg(char * message)
 {
   printf("%s\n",message);
   printf("Error code is %d\n",CSSM_GetError()->error);
   return(CSSM_FAIL);
 }

/*********************************************************************
 *  name: buildCertGroup - Allocate and load certificate data        *
 *********************************************************************/

 CSSM_RETURN buildCertGroup(CSSM_CERTGROUP * certGroupPtr,
                            char * certFile[], uint32 certCount)
 {
   FILE      * inFile;
   CSSM_DATA * certArray = (CSSM_DATA *) calloc(certCount,sizeof(CSSM_DATA));
   uint32    i, certSize;

   certGroupPtr->NumCerts = certCount;
   certGroupPtr->CertList = certArray;

   for (i=0; i <= certCount-1; i++)
   {
      inFile = fopen(certFile[i],"rb");
      if (!inFile)
      {
         printf("File %s could not be opened\n",certFile[i]);
         return(CSSM_FAIL);
      }
      /* Find size of certificate file */
      fseek(inFile,0L,SEEK_END);
      certSize = ftell(inFile);
      rewind(inFile);

      /* Read in certificate data*/
      certArray[i].Length = certSize;
      certArray[i].Data   = (uint8 *)calloc(certSize, sizeof(char));
      fread(certArray[i].Data, 1, certSize, inFile);
      fclose(inFile);
   }
   return(CSSM_OK);
 }
```

Figure 2. Example Code Using the OCEP Trust Policy APIs (Part 3 of 5)

```
/**********************************************************************
 *  name: freeCertGroup - Free certificate data storage              *
 *********************************************************************/

 void freeCertGroup(CSSM_CERTGROUP * certGroupPtr)
 {
   CSSM_DATA      * certArray = certGroupPtr->CertList;
   uint32           i;
   uint32           certCount = certGroupPtr->NumCerts;

   for (i=0; i <= certCount-1; i++)
   {
      free(certArray[i].Data);
   }
   free(certArray);
   return;
 }

/**********************************************************************
 *  name: openDB - Initialize data library                           *
 *********************************************************************/

 CSSM_RETURN openDB(char * ringName)
 {
   CSSM_DB_ACCESS_TYPE access = {CSSM_TRUE,CSSM_FALSE,CSSM_FALSE,CSSM_FALSE};

   dl_db_handle.DLHandle = ibm_dl_handle;
   dl_db_handle.DBHandle = CSSM_DL_DbOpen(ibm_dl_handle,
                                         ringName,
                                         &access,
                                         NULL,
                                         NULL);
   if (!dl_db_handle.DBHandle)
     return(errorMsg("Failed CSSM_DL_DbOpen"));

   ibm_dl_db_list.NumHandles = 1;
   ibm_dl_db_list.DLDBHandle = &dl_db_handle;

   return(CSSM_OK);
 }
/**********************************************************************
 *  name: closeDB - Free data library storage                        *
 *********************************************************************/

 void closeDB(void)
 {
   if (dl_db_handle.DBHandle)
     if (CSSM_DL_DbClose(dl_db_handle) != CSSM_OK)
       errorMsg("Failed CSSM_DL_DbClose");
   return;
 }
```

Figure 2. Example Code Using the OCEP Trust Policy APIs (Part 4 of 5)

```
/*********************************************************************
 *  name: attachPlugins - Attach required service provider modules   *
 ********************************************************************/

 CSSM_RETURN attachPlugins(void)
 {
   CSSM_GUID     ibmcsp_guid = IBMSWCSP_GUID;
   CSSM_VERSION  CL_version  = {IBM_CL_MAJOR_VERSION,   IBM_CL_MINOR_VERSION};
   CSSM_VERSION  CSP_version = {IBMSWCSP_MAJOR_VERSION, IBMSWCSP_MINOR_VERSION};
   CSSM_VERSION  TP_version  = {IBMOCEPTP_MAJOR_VERSION,IBMOCEPTP_MINOR_VERSION};
   CSSM_VERSION  DL_version; /* C compiler disallows DL version as initializer */
                 DL_version.Major = IBMOCEPDL_MAJOR_VERSION;
                 DL_version.Minor = IBMOCEPDL_MINOR_VERSION;
   ibm_cl_handle = CSSM_ModuleAttach(&ibmcl_guid, &CL_version,
                                     &MemoryFuncs, 0, 0, 0, NULL, NULL);
   if (!ibm_cl_handle) return(errorMsg("Failed attach of CL"));

   ibm_csp_handle = CSSM_ModuleAttach(&ibmcsp_guid, &CSP_version,
                                      &MemoryFuncs, 0, 0, 0, NULL, NULL);
   if (!ibm_csp_handle) return(errorMsg("Failed attach of CSP"));

   ibm_dl_handle = CSSM_ModuleAttach(&IBMOCEPDL_GUID, &DL_version,
                                     &MemoryFuncs, 0, 0, 0, NULL, NULL);
   if (!ibm_dl_handle) return(errorMsg("Failed attach of DL"));

   ibm_tp_handle = CSSM_ModuleAttach(&IBMOCEPTP_GUID, &TP_version,
                                     &MemoryFuncs, 0, 0, 0, NULL, NULL);
   if (!ibm_tp_handle) return(errorMsg("Failed attach of TP"));

   return(CSSM_OK);
 }

/*********************************************************************
 *  name: detachPlugins - Detach service provider modules             *
 ********************************************************************/

 void detachPlugins(void)
 {
   if (ibm_cl_handle)
     if (CSSM_ModuleDetach(ibm_cl_handle) != CSSM_OK)
       errorMsg("Failed detach of CL");

   if (ibm_csp_handle)
     if (CSSM_ModuleDetach(ibm_csp_handle) != CSSM_OK)
       errorMsg("Failed detach of CSP");

   if (ibm_dl_handle)
     if (CSSM_ModuleDetach(ibm_dl_handle) != CSSM_OK)
       errorMsg("Failed detach of DL");

   if (ibm_tp_handle)
     if (CSSM_ModuleDetach(ibm_tp_handle) != CSSM_OK)
       errorMsg("Failed detach of TP");

   return;
 }
```

Figure 2. Example Code Using the OCEP Trust Policy APIs (Part 5 of 5)

Chapter 4. Using Data Storage Library Services

This section describes the OCEP Data Storage Library service provider module. It also describes its implementation of the data library APIs that are defined in the OCSF Framework.

Using the Data Storage Library Services Module

After you complete the installation steps described in "Installing the OCEP Code" on page 10, the following OCEP Data Storage Library service provider files are available for use on your system.

Function	Name	Directory Location
Header File	ibmocepdl.h	`/user/lpp/ocsf/include`
Executable Module	ibmocepdl.so	`/user/lpp/ocsf/addins`

To use the OCEP Data Storage Library Services, an application must explicitly attach this service provider module. To do so, the application must use CSSM_ModuleAttach, which is an API provided by OCSF, to attach the specific GUID for the service provider module. In turn, CSSM_ModuleAttach returns a handle that uniquely represents the pairing of the OCEP service provider module and the calling application.

The following GUID identifies the OCEP Data Storage Library service provider module; this GUID and other related constants are defined in the ibmocepdl.h header file:

```
// {5E43B2A3-1C38-11d2-8688-0004ACF320BC}
static const CSSM_GUID IBMOCEPDL_GUID =
{ 0x5e43b2a3, 0x1c38, 0x11d2, { 0x86, 0x88, 0x0, 0x4, 0xac, 0xf3, 0x20, 0xbc } };
```

For more information about the CSSM_ModuleAttach function and developing security applications, see the *z/OS Open Cryptographic Services Facility Application Programming*.

Supported Data Library APIs

Table 5 summarizes the data library functions that are defined in the OCSF Framework and if they are supported by the OCEP Data Storage Library service provider module.

Table 5. Data Storage Library Functions that are Supported by OCEP

Function Name	Supported	Comments
CSSM_DL_AbortQuery	Yes	See page 25.
CSSM_DL_Authenticate	No	
CSSM_DL_DbClose	Yes	See page 26.
CSSM_DL_DbCreate	No	
CSSM_DL_DbDelete	No	
CSSM_DL_DbExport	No	
CSSM_DL_DbImport	No	

Table 5. Data Storage Library Functions that are Supported by OCEP (continued)

Function Name	Supported	Comments
CSSM_DL_DbOpen	Yes	See page 27.
CSSM_DL_DbSetRecordParsingFunctions	No	
CSSM_DL_DbGetRecordParsingFunctions	No	
CSSM_DL_GetDbNameFromHandle	No	
CSSM_DL_DataDelete	No	
CSSM_DL_DataGetFirst	Yes	See page 28.
CSSM_DL_DataGetNext	Yes	See page 31.
CSSM_DL_DataInsert	No	
CSSM_DL_FreeUniqueRecord	Yes	See page 33.
CSSM_DL_PassThrough	No	

Note: The following sections provide an overview of the APIs that are supported by the OCEP Data Storage Library service provider modules. Only those parameters and values that are unique to OCEP's implementation are described.

For a full description of the syntax and supporting parameters of the remaining APIs that are implemented in the OCSF Framework, refer to the *z/OS Open Cryptographic Services Facility Application Programming*.

CSSM_DL_AbortQuery

Description

This function ends the query that was initiated by CSSM_DL_DataGetFirst or CSSM_DL_DataGetNext. It releases the *ResultsHandle* that was returned by a previous query. The calling application must use this API to free the related storage that was obtained.

Format

```
CSSM_RETURN  CSSMAPI  CSSM_DL_DataAbortQuery
        (CSSM_DL_DB_HANDLE  DLDBHandle,
         CSSM_HANDLE  ResultsHandle)
```

Parameters

DLDBHandle **(input)**
specifies the RACF key ring handle; this is a required value.

ResultsHandle **(input)**
is the handle returned by the CSSM_DL_DataGetFirst function.

CSSM_DL_DbClose

Description

This function closes an open key ring (data store). The calling application must use this API to free the related storage that was obtained.

Format

```
CSSM_RETURN  CSSMAPI  CSSM_DL_DbClose
        (CSSM_DL_DB_HANDLE  DLDBHandle)
```

Parameters

DLDBHandle **(input)**

specifies the RACF key ring handle; this is a required value.

CSSM_DL_DbOpen

Description

This function opens the specified key ring (data store).

Format

```
CSSM_DB_HANDLE CSSMAPI CSSM_DL_DbOpen
     (CSSM_DL_HANDLE DLHandle,
       const char *DbName,
       const CSSM_DB_ACCESS_TYPE_PTR AccessRequest,
       const CSSM_USER_AUTHENTICATION_PTR UserAuthentication,
       const void *OpenParameters)
```

Parameters

DLHandle **(input)**
the handle that describes the data storage library module to be used to perform this function.

DbName **(input)**
a pointer to the string containing the logical name of the key ring (data store). This name has the following format:

```
userid/user-key ring
```

userid the 1-8 character user ID associated with this key ring; the user ID must be specified in uppercase characters.

user-key ring
the case-sensitive ring name, which may contain up to 237 characters

AccessRequest **(input)**
indicates the requested access mode; this must be specified as READONLY.

UserAuthentication **(input)**
must be specified as NULL, as RACF access controls will be used to determine user authentication.

OpenParameters **(input)**
this parameter is ignored and must be specified as NULL.

CSSM_DL_DataGetFirst

Description

This function retrieves the first record in the key ring (data store) that matches the given selection criteria. Information is only returned for certificates that have been marked as trusted by RACF. If the certificate has not been marked as trusted, it is not returned to the application; that is, it is as if the certificate is not connected to the key ring.

The selection criteria is specified in the *Query* structure, which has specific characteristics when used with the OCEP Data Storage Library service provider module. The function returns a unique record identifier that is associated with the retrieved record. This identifier can then be used in other references to the retrieved data record. For example, it can be specified on calls to CSSM_DL_FreeUniqueRecord.

Notes:

1. The calling application is responsible for freeing the storage that is acquired for the returned *Data* (including its sub-pieces CertData and PvtKeyData) and *Attributes* parameters. Also, the storage that was acquired for the CSSM_DB_UNIQUE_RECORD must be freed by calling CSSM_DL_FreeUniqueRecord. In addition, the storage that was acquired for the results handle must be freed by calling CSSM_DL_AbortQuery.
2. Because the private key data returned could be either an ICSF token label or a non-ICSF key, the application must attach the appropriate Cryptographic Service Provider (CSP) as identified by the *CspId* field in the CSSM_KEYHEADER.

Format

```
CSSM_DB_UNIQUE_RECORD_PTR  CSSMAPI  CSSM_DL_DataGetFirst
     (CSSM_DL_DB_HANDLE  DLDBHandle,
      const  CSSM_QUERY_PTR  Query,
      CSSM_HANDLE_PTR  ResultsHandle,
      CSSM_BOOL  *EndOfDataStore,
      CSSM_DB_RECORD_ATTRIBUTE_DATA_PTR  Attributes,
      CSSM_DATA_PTR  Data)
```

Parameters

DLDBHandle **(input)**
specifies the RACF key ring handle; this is a required value.

Query **(input)**
specifies the information that will be used to query the specified key ring; this is a required value and it must have the following structure:

RecordType
must be set to CSSM_DL_DB_RECORD_CERT.

Conjuntive
must be set to CSSM_DB_NONE.

NumSelectionPredicates
must be either 0 or 1. If set to 1, then *SelectionPredicates* must point to a CSSM_SELECTION_PREDICATE structure, which has the following format:

DbOperator
must be set to CSSM_DB_EQUAL

Attribute
one of the queriable attributes, coded as follows:

- *Info.AttributeNameFormat*, which must be set to CSSM_DB_ATTRIBUTE_NAME_AS_NUMBER
- *Info.Label.AttributeNumber*, which must be one the following vales:
  - CSSM_DL_ATTRIBUTE_LABEL = 0x3 - Query on label
  - OCEP_DL_ATTRIBUTE_DEFAULT = 0x4 - Query on default flag (this constant is defined in the ibmocepdl.h header file)
  - CSSM_DL_ATTRIBUTE_SUBJECT = 0x101 - Query on DER-encoded subject's name

ResultsHandle **(output)**
contains the key ring handle, which should be saved and used to retrieve subsequent records that satisfied this query.

EndOfDataStore **(output)**
one of the following flags, which indicates if a record that satisfied this query was available to be retrieved in the current operation:

CSSM_FALSE
a record was available and was retrieved, unless an error condition occurred.

CSSM_TRUE
all records satisfying the query have been previously retrieved and no record has been returned by this operation.

Attributes **(output)**
contains the attribute values of the retrieved record. This structure has the following format:

SemanticInformation
a structure defined by CSSM_DB_CERTRECORD_SEMANTICS; the following flags are supported:

- CSSM_DB_CERT_USE_TRUSTED, which indicates this is a Certificate Authority certificate.
- CSSM_DB_CERT_USE_OWNER, which indicates this is User/Server certificate, with a possible private key.

If neither bit is set, a SITE certificate is indicated. A SITE certificate is one that the RACF administrator has explicitly defined and added as a trusted certificate.

NumberOfAttributes
indicates the number of CSSM_DB_ATTRIBUTE_DATA structures that are pointed to by *Attributes*. Each of these structures will be coded as the *Query* attribute, as described on page 28. In addition, the following non-queriable attribute will also be present:

- CSSM_DL_ATTRIBUTE_ID = 0x101 - The RACF user ID that is associated with this certificate profile

Data **(output)**
is a pointer to a CSSM_DATA structure that contains the nonattribute record data; for RACF, this is the certificate and an optional private key. Data->Data will point to the following structure:

```
typedef struct ocep_cert_key_record {
 CSSM_DATA CertData;    //DER encoded certificate
 CSSM_KEY PrvtKeyData;  //Optional Private key,
                        //KeyData.Length=KeyData.Data=NULL if not present
} OCEP_CERT_KEY_RECORD, *OCEP_CERT_KEY_RECORD_PTR
```

CSSM_DL_DataGetNext

Description

This function retrieves the next data record in the key ring that matches the selection criteria (specified by the CSSM_DL_DataGetFirst function). Information is only returned for certificates that have been marked as trusted by RACF; if the certificate has not been marked as trusted, it will not be returned to the calling application.

Format

```
CSSM_DB_UNIQUE_RECORD_PTR  CSSMAPI  CSSM_DL_DataGetNext
        (CSSM_DL_DB_HANDLE  DLDBHandle,
          CSSM_HANDLE  ResultsHandle,
          CSSM_BOOL  *EndOfDataStore,
          CSSM_DB_RECORD_ATTRIBUTE_DATA_PTR  Attributes,
          CSSM_DATA_PTR  Data)
```

Parameters

DLDBHandle **(input)**
specifies the RACF key ring handle; this is a required value.

ResultsHandle **(input)**
this is the handle that is returned by the CSSM_DL_DataGetFirst function.

EndOfDataStore **(output)**
one of the following flags, which indicates if a record that satisfied this query was available to be retrieved in the current operation:

CSSM_FALSE
a record was available and was retrieved, unless an error condition occurred.

CSSM_TRUE
all records satisfying the query have been previously retrieved and no record has been returned by this operation.

Attributes **(output)**
contains the attribute values of the retrieved record. This structure has the following format:

SemanticInformation
a structure defined by CSSM_DB_CERTRECORD_SEMANTICS; the following flags are supported:

- CSSM_DB_CERT_USE_TRUSTED, which indicates this is a Certificate Authority certificate.
- CSSM_DB_CERT_USE_OWNER, which indicates this is a User/Server certificate, with a possible private key.

If neither bit is set, a SITE certificate is indicated. A SITE certificate is one that the RACF administrator has explicitly defined and added as a trusted certificate.

NumberOfAttributes
indicates the number of CSSM_DB_ATTRIBUTE_DATA structures that are pointed to by *Attributes*. Each of these structure will be coded as the *Query* attribute, as described on page 28. In addition, the following non-queriable attribute will also be present:

- CSSM_DL_ATTRIBUTE_ID = 0x101 - The RACF user ID that is associated with this certificate profile.

Data **(output)**

is a pointer to a CSSM_DATA structure that contains the nonattribute record data; for RACF, this is the certificate and an optional private key. Data->Data will point to the following structure:

```
typedef struct ocep_cert_key_record {
 CSSM_DATA CertData;    //DER encoded certificate
 CSSM_KEY PrvtKeyData;  //Optional Private key,
                        //KeyData.Length=KeyData.Data=NULL if not present
} OCEP_CERT_KEY_RECORD, *OCEP_CERT_KEY_RECORD_PTR
```

CSSM_DL_FreeUniqueRecord

Description

Frees the pointer to the unique record ID that is returned by the CSSM_DL_DataGetFirst or CSSM_DL_DataGetNext functions. The record itself and the data it contains are unchanged. The calling application must use this API to free the related storage that was obtained.

Format

```
CSSM_RETURN CSSMAPI CSSM_DL_FreeUniqueRecord
        (CSSM_DL_DB_HANDLE DLDBHandle,
         CSSM_DB_UNIQUE_RECORD_PTR UniqueRecord)
```

Parameters

DLDBHandle **(input)**
specifies the RACF key ring handle; this is a required value.

UniqueRecord **(input)**
the unique record ID (CSSM_DB_UNIQUE_RECORD), which is returned by CSSM_DL_DataGetFirst or CSSM_DL_DataGetNext.

Error Codes

Table 6 lists the error codes that are unique to OCEP's support of the Data Storage Library Services APIs.

Table 6. OCEP Data Storage Library Error Codes

Decimal Value	Error Description
6010	Number of selection predicates exceeded the maximum
6011	Incorrect attribute length was specified
6012	Incorrect *DBName* specified
6013	Incorrect user ID length specified
6014	Ring name is missing
6015	Unsupported access request type
6016	SAF service (IRRSDL00) not available
6800 - 6899	Errors that are returned by the IRRSDL00 (R_datalib) callable service. The last two decimal digits represent the reason code returned from the service. See the *z/OS Security Server RACF Callable Services* book for more information about these error codes.

For information about the OCSF APIs that perform error reporting and recovery, plus a list of other OCSF-defined error codes, refer to the *z/OS Open Cryptographic Services Facility Application Programming*.

Data Storage Library Example

Figure 3 on page 35 shows excerpts from a sample program that uses the data storage library APIs that are supported by OCEP; this is not a complete program. For an example of how to attach the OCEP Data Storage Library service provider module, see the sample program in Figure 2 on page 17.

The **highlighted** entries, however, demonstrate how you could use the supported APIs to extract the default certificate and private key from a key ring called "MyRing". The key ring is owned by user ID WEBSRVR. This example also returns the DER-encoded subject's distinguished name.

```
#include "ibmocepdl.h"

/* Declare the key ring info */
CSSM_DL_DB_HANDLE Handles;
CSSM_DB_ACCESS_TYPE READONLY = { CSSM_TRUE, CSSM_FALSE, CSSM_FALSE, CSSM_FALSE };
char ringname[] = "WEBSRVR/MyRing";

/* Declare one attribute to search on, DEFAULT*/
CSSM_SELECTION_PREDICATE DefFlag;
CSSM_QUERY MyQuery;
int YES = 1;

/* Declare the output fields */
CSSM_DB_UNIQUE_RECORD_PTR Record_ID;
CSSM_HANDLE OutScanHandle;
CSSM_BOOL EOData;
CSSM_DB_RECORD_ATTRIBUTE_DATA OutAttributes;
OCEP_CERT_KEY_RECORD *MyCertAndKey;
CSSM_DATA OutData, MyCert, MySubjectsName;
CSSM_KEY MyKey;

/* Declare misc */
CSSM_DB_ATTRIBUTE_DATA_PTR p;
int i;

/* Open the key ring. This assumes the OCEP DL has already been attached
and Handles.DLHandle set  */
Handles.DBHandle=
CSSM_DL_DbOpen(Handles.DLHandle,ringname,READONLY,NULL,NULL);
```

Figure 3. Example Code Using the OCEP Data Storage Library Services APIs (Part 1 of 2)

```
/* Setup the attribute value */
DefFlag.DbOperator= CSSM_DB_EQUAL;
DefFlag.Attribute.Value.Length=Size_Of(YES);  // Length must be four bytes
DefFlag.Attribute.Value.Data= &YES;
DefFlag.Attribute.Info.AttributeNameFormat=
CSSM_DB_ATTRIBUTE_NAME_AS_NUMBER;
DefFlagAttribute.Info.Label.AttributeNumber= OCEP_DL_ATTRIBUTE_DEFAULT;

/* Prepare the query */
MyQuery.RecordType= CSSM_DL_DB_RECORD_CERT;
MyQuery.Conjunctive= CSSM_DB_NONE;
MyQuery.NumSelectionPredicates= 1;
MyQuery.SelectionPredicate= &DefFlag;

Record_ID=
CSSM_DL_DataGetFirst(Handles,&MyQuery,&OutScanHandle,&EOData,&OutAttributes,&OutData);
if (!EOData && Record_ID) // If record returned
{
   /* Get the DER encoded certificate */
   MyCertAndKey= OutData.Data;   // Data points to an OCEP_CERT_KEY_RECORD
   MyCert.Length= MyCertAndKey->CertData.Length;  // Length of DER encoded certificate
   MyCert.Data= MyCertAndKey->CertData.Data;  // DER encoded certificate
   if (MyCertAndKey->PrvtKeyData.KeyData.Length != 0)  // Is a private key present?
   {
        /* Get the private key */
        MyKey.KeyData.Length= MyCertAndKey->PrvtKeyData.KeyData.Length;
        MyKey.KeyData.Data= MyCertAndKey->PrvtKeyData.KeyData.Data;
         memcpy(MyKey.KeyHeader,
           MyCertAndKey->PrvtKeyData.KeyHeader,sizeof(CSSM_KEYHEADER);
   }
   else
       ;  // perform some error action
   /* Get the subject's DN */
   for (i=0,p=OutAttributes.AttributeData ; i < OutAttributes.NumberOfAttributes ; i++,p++)
        if (p->Info.Label.AttributeNumber == CSSM_DL_ATTRIBUTE_SUBJECT)
        {
           MySubjectsName.Length= p->Value.Length;
           MySubjectsName.Data= p->Value.Data;
        }
   //
   // Make use of the certificate/key/subject's name here
   //
   /* Clean up this record */
   free(MyCertAndKey->CertData.Data); // Free certificate storage
   free(MyCertAndKey->PrvtKeyData.KeyData.Data); // Free key data storage
   free(MyCertAndKey);  // Free OCEP_CERT_KEY_RECORD storage
   /* Now clean up the attributes */
   for (i=0,p=OutAttributes.AttributeData ; i < OutAttributes.NumberOfAttributes ; i++,p++)
      free(p->Value.Data); // Free individual attribute data
   free(OutAttributes.AttributeData); // Free CSSM_DB_ATTRIBUTE_DATA list
   CSSM_DL_FreeUniqueRecord(Handles,Record_ID); // Free storage associated with the record ID
}
/* Cleanup this key ring scan */
CSSM_DL_AbortQuery(Handles,OutScanHandle);
/* Close the key ring */
CSSM_DL_DbClose(Handles);
```

Figure 3. Example Code Using the OCEP Data Storage Library Services APIs (Part 2 of 2)

Notices

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact: but do not indicate that it is the legal department.

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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

COPYRIGHT LICENSE:

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

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This publication documents intended Programming Interfaces that allow customers to write programs to obtain services of Open Cryptographic Enhanced Plug-ins (OCEP).

Trademarks

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

BookManager	IBM	IBMLink
Language Environment	OS/390	RACF

z/OS

UNIX is a registered trademark in the United States, other countries, or both and is licensed exclusively through X/Open Company Limited.

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

Index

Readers' Comments — We'd Like to Hear from You

z/OS
Integrated Security Services
Open Cryptographic Enhanced Plug-ins
Application Programming

Publication No. SC24-5925-01

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: mhvrcfs@us.ibm.com

If you would like a response from IBM, please fill in the following information:

Name ________________________

Address ________________________

Company or Organization ________________________

____

Phone No. ________________________

E-mail address ________________________

Readers' Comments — We'd Like to Hear from You
SC24-5925-01

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 Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY
12601-5400

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Program Number: 5694-A01

Printed in USA

SC24-5925-01