z/OS

Integrated Security Services Enterprise Identity Mapping (EIM) Guide and Reference

SA22-7875-07

z/OS

Integrated Security Services Enterprise Identity Mapping (EIM) Guide and Reference

SA22-7875-07

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 463.

Eighth Edition, September 2008

This edition applies to Version 1 Release 10 of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

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

International Business Machines Corporation
MHVRCFS, Mail Station P181
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405
FAX (Other Countries):
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/systems/z/os/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 document
- 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

Tables

Figures

About this document

This document supports z/OS® (5694-A01). This document contains information about using Enterprise Identity Mapping (EIM). EIM is an architecture that serves as a security technology to make it easier to manage users in a cross-platform environment. For a detailed introduction of EIM, see Chapter 1, “Enterprise Identity Mapping (EIM),” on page 3.

Who should use this document

EIM requires a Lightweight Directory Access Protocol (LDAP) server because EIM data is stored in LDAP. (For information about LDAP requirements, see “Planning considerations for an EIM domain controller” on page 46.) EIM optionally requires RACF® or an equivalent external security manager.

Those who might find this document helpful include the following:

- EIM administrators who plan, install, configure, customize, administer, or use EIM
- RACF administrators who issue commands in support of EIM
- Application programmers
- LDAP administrators who install, configure, or administer LDAP in support of EIM

(See “Team members” on page 37 for a complete list of team members.)

This document assumes that you are familiar with the following concepts and protocols:

- LDAP
- z/OS UNIX® System Services shell
- C/C++ programming languages

How to use this document

For a detailed introduction of EIM, see Chapter 1, “Enterprise Identity Mapping (EIM),” on page 3.

Where to find more information

Where necessary, this book refers to information in other books. For complete titles and order numbers for all elements of z/OS, see *z/OS Information Roadmap*.

Softcopy publications

The Security Server library is available on the following CD-ROM, DVD, and online library collections. The collections include the IBM® Sofctcopy Reader, which is a program that enables you to view the softcopy documents.

SK3T-4269 *z/OS Version 1 Release 10 Collection*

This collection contains the set of unlicensed documents for the current release of z/OS in both BookManager® and Portable Document Format (PDF) files. You can view or print the PDF files with the Adobe® Acrobat reader.

SK3T-4272 *z/OS Security Server RACF Collection*

This softcopy collection kit contains the z/OS Security Server library in both BookManager and Portable Document Format (PDF) files. You can view or print the PDF files with the Adobe Acrobat reader.

SK3T-4271 *z/OS Version 1 Release 10 and Software Products DVD Collection*

This DVD collection contains libraries for a single release of z/OS, plus libraries for multiple releases of related software products that run on z/OS. It also includes selected IBM Redbooks®. The documents are provided in both BookManager and PDF formats when available.

Other sources of information

IBM provides customer-accessible discussion areas where EIM and RACF may be discussed by customer and IBM participants. Other information is also available through the Internet.

Internet sources

EIM is a cross-platform infrastructure that is available on the following platforms:

- *iSeries*
- *pSeries*
- *xSeries*
- *zSeries*

The following resources are available through the Internet to provide additional information about EIM and other security-related topics:

- **EIM home page**

  Visit the EIM Web page:

  `www.ibm.com/servers/eserver/security/eim`
- **Online library**

  To view and print online versions of the z/OS publications, use this address:

  `http://www.ibm.com/systems/z/os/zos/bkserv/`
- **Redbooks**

  The Redbooks that are produced by the International Technical Support Organization (ITSO) are available at the following address:

  `http://www.redbooks.ibm.com`
- **Enterprise systems security**

  For more information about security on the S/390® platform, OS/390®, and z/OS, including the elements that comprise the Security Server, use this address:

  `http://www.ibm.com/systems/z/advantages/security/`
- **RACF home page**

  You can visit the RACF home page on the World Wide Web using this address:

  `http://www.ibm.com/servers/eserver/zseries/zos/racf/`
- **RACF-L discussion list**

  Customers and IBM participants can also discuss RACF on the RACF-L discussion list. RACF-L is not operated or sponsored by IBM. It is run by the University of Georgia.

  To subscribe to the RACF-L discussion and receive postings, send a note to:

  `listserv@listserv.uga.edu`

Include the following line in the body of the note, substituting your first name and last name as indicated:

```
subscribe racf-l first_name last_name
```

To post a question or response to RACF-L, send a note, including an appropriate `Subject:` line, to:

```
racf-l@listserv.uga.edu
```

- **Sample code**

  You can get sample code, internally-developed tools, and exits to help you use RACF. This code works in IBM's test environment, at the time we make it available, but is not officially supported. Each tool or sample has a README file that describes the tool or sample and any restrictions on its use.

  To access this code from a Web browser, go to the RACF home page and select the "Downloads" topic from the navigation bar, or go to ftp://ftp.software.ibm.com/eserver/zseries/zos/racf/.

  The code is also available from `ftp.software.ibm.com` through anonymous FTP. To get access:

  1. Log in as user **anonymous**.
  2. Change the directory, as follows, to find the subdirectories that contain the sample code or tool you want to download:

     ```
     cd eserver/zseries/zos/racf/
     ```

  An announcement is posted on RACF-L, MVSRACF, and SECURITY CFORUM whenever function is added.

  Note: Some Web browsers and some FTP clients (especially those using a graphical interface) might have problems using `ftp.software.ibm.com` because of inconsistencies in the way they implement the FTP protocols. If you have problems, you can try the following:

  – Try to get access by using a Web browser and the links from the RACF home page.
  – Use a different FTP client. If necessary, use a client that is based on command line interfaces instead of graphical interfaces.
  – If your FTP client has configuration parameters for the type of remote system, configure it as UNIX instead of MVS™.

Restrictions

Because the sample code and tools are not officially supported,

– There are no guaranteed enhancements.
– No APARs can be accepted.

To request copies of IBM publications

Direct your request for copies of any IBM publication to your IBM representative or to the IBM branch office serving your locality.

There is also a toll-free customer support number (1-800-879-2755) available Monday through Friday from 8:30 a.m. through 5:00 p.m. Eastern Time. You can use this number to:

- Order or inquire about IBM publications
- Resolve any software manufacturing or delivery concerns

- Activate the program reorder form to provide faster and more convenient ordering of software updates

Summary of changes

**Summary of changes
for SA22–7875–07
z/OS Version 1 Release 10**

This document contains information previously presented in z/OS Integrated Security Services Enterprise Identity Mapping (EIM) Guide and Reference, SA22-7875-06, which supports z/OS Version 1 Release 8. The following summarizes the changes to that information.

Updated information

This release of EIM updates the following information:

- "Configuring the IBM Tivoli Directory Server for remote services support" on page 432 updated to show new content of section in the Tivoli® Directory Server SLAPDCNF configuration file to enable ICTX extended operations.
- "Remote authorization requests" on page 449 updated to include information on RACF group authorization requests.

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.

**Summary of changes
for SA22–7875–06
z/OS Version 1 Release 8**

This document contains information previously presented in z/OS Integrated Security Services Enterprise Identity Mapping (EIM) Guide and Reference, SA22-7875-05, which supports z/OS Version 1 Release 7. The following summarizes the changes to that information.

New information

This release of EIM introduces the following:

- A new section, "Remote Services" has been added.
- New chapters detailing working with identity cache and remote authorization and auditing have been added to this new section. See Chapter 13, "The z/OS Identity Cache," on page 423, Chapter 14, "ICTX Java API," on page 431, and Chapter 15, "Accessing RACF remotely to perform authorization checks and create audit records," on page 447 for more details.

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.

**Summary of changes
for SA22–7875–05
z/OS Version 1 Release 7**

This document contains information previously presented in z/OS Integrated Security Services Enterprise Identity Mapping (EIM) Guide and Reference,

SA22-7875-04, which supports z/OS Version 1 Release 7. The following summarizes the changes to that information.

Updated information

This release of EIM updates the following information:

- More detailed explanations of working with audit records.
- A thorough description of working with the tabular output created by the SMF Unload utility.

See "Working with audit records" on page 142 for these updates.

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.

Summary of changes
for SA22–7875–04
z/OS Version 1 Release 7

This document contains information previously presented in z/OS Integrated Security Services Enterprise Identity Mapping (EIM) Guide and Reference, SA22-7875-02, which supports z/OS Version 1 Release 6. The following summarizes the changes to that information.

New information

This release of EIM introduces the following:

- Auditing support for EIM events. For more details, see Chapter 10, "EIM Auditing," on page 137.
- The addition of Java™ APIs, which make EIM available to z/OS applications and servers written in Java. For more details see "Java APIs" on page 159.

Updated information

This release of EIM updates the following information:

- Chapter 9, "The eimadmin utility," on page 109 has been updated to include the -U flag.
- The APF authorization requirement has been removed from EIM APIs. This important change requires a modification to your existing programs if you're migrating from a previous release of z/OS. For more details, see "Preparing to run an EIM application" on page 80.

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.

Summary of changes
for SA22–7875–03
z/OS Version 1 Release 7

This document contains information previously presented in z/OS Integrated Security Services Enterprise Identity Mapping (EIM) Guide and Reference, SA22-7875-02, which supports z/OS Version 1 Release 6. The following summarizes the changes to that information.

New information

This release of EIM introduces the following:

- Auditing support for EIM events. For more details, see Chapter 10, “EIM Auditing,” on page 137.
- The addition of Java APIs, which make EIM available to z/OS applications and servers written in Java. For more details see “Java APIs” on page 159.

Updated information

This release of EIM updates the following information:

- Chapter 9, “The eimadmin utility,” on page 109 has been updated to include the -U flag.
- The APF authorization requirement has been removed from EIM APIs. This important change requires a modification to your existing programs if you’re migrating from a previous release of z/OS. For more details, see “Preparing to run an EIM application” on page 80.

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.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of changes for SA22–7875–02 z/OS Version 1 Release 6

This document contains information previously presented in z/OS Security Server Enterprise Identity Mapping (EIM) Guide and Reference, SA22-7875-01, which supports z/OS Version 1 Release 5. The following summarizes the changes to that information.

New information

This release of EIM introduces the following:

- The use of policies, which are default mappings for registries and an EIM domain
- New APIs to support the use of policies
- Support for X.509 certificate registries

Updated information

This release of EIM updates the following information:

- The conceptual information in Chapters 1 and 2, including artwork, has been updated to support policies information
- The eimadmin utility chapter has been updated

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.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of changes
for SA22-7875-01
z/OS Version 1 Release 5

This document contains information previously presented in z/OS Security Server Enterprise Identity Mapping (EIM) Guide and Reference, SA22-7875-00, which supports z/OS Version 1 Release 4.The following summarizes the changes to that information.

New information

- Additional bind mechanism support was added to allow applications and administrators to bind to LDAP with a Kerberos credential or digital certificate. In addition, CRAM-MD5 password protection is now supported for simple bind credentials.

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.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Part 1. EIM concepts and use

Chapter 1. Enterprise Identity Mapping (EIM)

Today's network environments are made up of a complex group of systems and applications, resulting in the need to manage multiple user registries. Dealing with multiple user registries quickly grows into a large administrative problem that affects users, administrators, and application developers. Consequently, many companies are struggling to securely manage authentication and authorization for systems and applications. Enterprise Identity Mapping (EIM) is an IBM eserver infrastructure technology that allows administrators and application developers to address this problem more easily and inexpensively than previously possible.

The following information describes the problems, outlines current industry approaches, and explains why the EIM approach is better.

The problem: Managing multiple user registries

Many administrators manage networks that include different systems and servers, each with a unique way of managing users through various user registries. In these complex networks, administrators are responsible for managing each user's identities and passwords across multiple systems. Additionally, administrators often must synchronize these identities and passwords and users are burdened with remembering multiple identities and passwords and with keeping them in sync. The user and administrator overhead in this environment is excessive. Consequently, administrators often spend valuable time troubleshooting failed logon attempts and resetting forgotten passwords instead of managing the enterprise.

The problem of managing multiple user registries also affects application developers who want to provide multiple-tier or heterogeneous applications. These developers understand that customers have important business data spread across many different types of systems, with each system possessing its own user registries. Consequently, developers must create proprietary user registries and associated security semantics for their applications. Although this solves the problem for the application developer, it increases the overhead for users and administrators.

Current approaches

Several current industry approaches for solving the problem of managing multiple user registries are available, but they all provide incomplete solutions. For example, Lightweight Directory Access Protocol (LDAP) provides a distributed user registry solution. However, using LDAP (or other popular solutions such as Microsoft® Passport) means that administrators must manage yet another user registry and set of security semantics or must replace existing applications that are built to use those registries.

Using this type of solution, administrators must manage multiple security mechanisms for individual resources, thereby increasing administrative overhead and potentially increasing the likelihood of security exposures. When multiple mechanisms support a single resource, the chance of changing the authority through one mechanism and forgetting to change the authority for one or more of the other mechanisms is much higher. For example, a security exposure can result when a user is appropriately denied access through one interface, but allowed access through one or more other interfaces.

After completing this work, administrators find that they have not completely solved the problem. Generally, enterprises have invested too much money in current user

registries and in their associated security semantics to make using this type of solution practical. Creating another user registry and associated security semantics solves the problem for the application provider, but not the problems for users or administrators.

One other possible solution is to use a single sign-on approach. Several products are available that allow administrators to manage files that contain all of a user's identities and passwords. However, this approach has several weaknesses:

- It addresses only one of the problems that users face. Although it allows users to sign on to multiple systems by supplying one identity and password, it does not eliminate the need for the user to have passwords on other systems, or the need to manage these passwords.
- It introduces a new problem by creating a security exposure because clear-text or decryptable passwords are stored in these files. Passwords should never be stored in in clear-text files or be easily accessible by anyone, including administrators.
- It does not solve the problems of third-party application developers who provide heterogeneous, multiple-tier applications. They must still provide proprietary user registries for their applications.

Despite these weaknesses, some enterprises have chosen to adopt these approaches because they provide some relief for the multiple user registry problems.

The EIM approach

EIM offers a new approach to enable inexpensive solutions to easily manage multiple user registries and user identities in an enterprise. EIM is an architecture for describing the relationships between individuals or entities (like file servers and print servers) in the enterprise and the many identities that represent them within an enterprise. In addition, EIM provides a set of APIs that allow applications to ask questions about these relationships.

For example, given a person's user identity in one user registry, you can determine which user identity in another user registry represents that same person. If the user has authenticated with one user identity and you can map that user identity to the appropriate identity in another user registry, the user does not need to provide credentials for authentication again. You know who the user is and only need to know which user identity represents that user in another user registry. Therefore, EIM provides a generalized identity mapping function for the enterprise.

EIM allows one-to-many mappings (in other words, a single user with more than one user identity in a single user registry). However, the administrator does not need to have specific individual mappings for all user identities in a user registry. EIM also allows many-to-one mappings (in other words, multiple users mapped to a single user identity in a single user registry).

The ability to map between a user's identities in different user registries provides many benefits. Primarily, it means that applications may have the flexibility of using one user registry for authentication while using an entirely different user registry for authorization. For example, an administrator could map an SAP identity (or better yet, SAP could do the mapping itself) to access SAP resources.

The use of identity mapping requires that administrators do the following:

1. Create EIM identifiers that represent people or entities in their enterprise

2. Create EIM registry definitions that describe the existing user registries in their enterprise
3. Define the relationship between the user identities in those registries to the EIM identifiers that they created
4. Create policy associations

No code changes are required to existing user registries. The administrator does not need to have mappings for all identities in a user registry. EIM allows one-to-many mappings (in other words, a single user with more than one user identity in a single user registry). EIM also allows many-to-one mappings (in other words, multiple users sharing a single user identity in a single user registry, which although supported is not advised). An administrator can represent any user registry of any type in EIM.

EIM is an open architecture that administrators may use to represent identity mapping relationships for any registry. It does not require copying existing data to a new repository and trying to keep both copies synchronized. The only new data that EIM introduces is the relationship information. Administrators manage this data in an LDAP directory, which provides the flexibility of managing the data in one place and having replicas wherever the information is used. Ultimately, EIM gives enterprises and application developers the flexibility to easily work in a wider range of environments with less cost than would be possible without this support.

Chapter 2. EIM concepts

A conceptual understanding of how Enterprise Identity Mapping (EIM) works is necessary to fully understand how you can use EIM in your enterprise. Although the configuration and implementation of EIM APIs can differ among server platforms, EIM concepts are common across IBM eServer™ servers.

The following figure provides an EIM implementation example in an enterprise. Three servers act as EIM clients and contain EIM-enabled applications that request EIM data using lookup operations. The domain controller stores information about the EIM domain, which includes an EIM identifier, associations between these EIM identifiers and user identities, and EIM registry definitions.

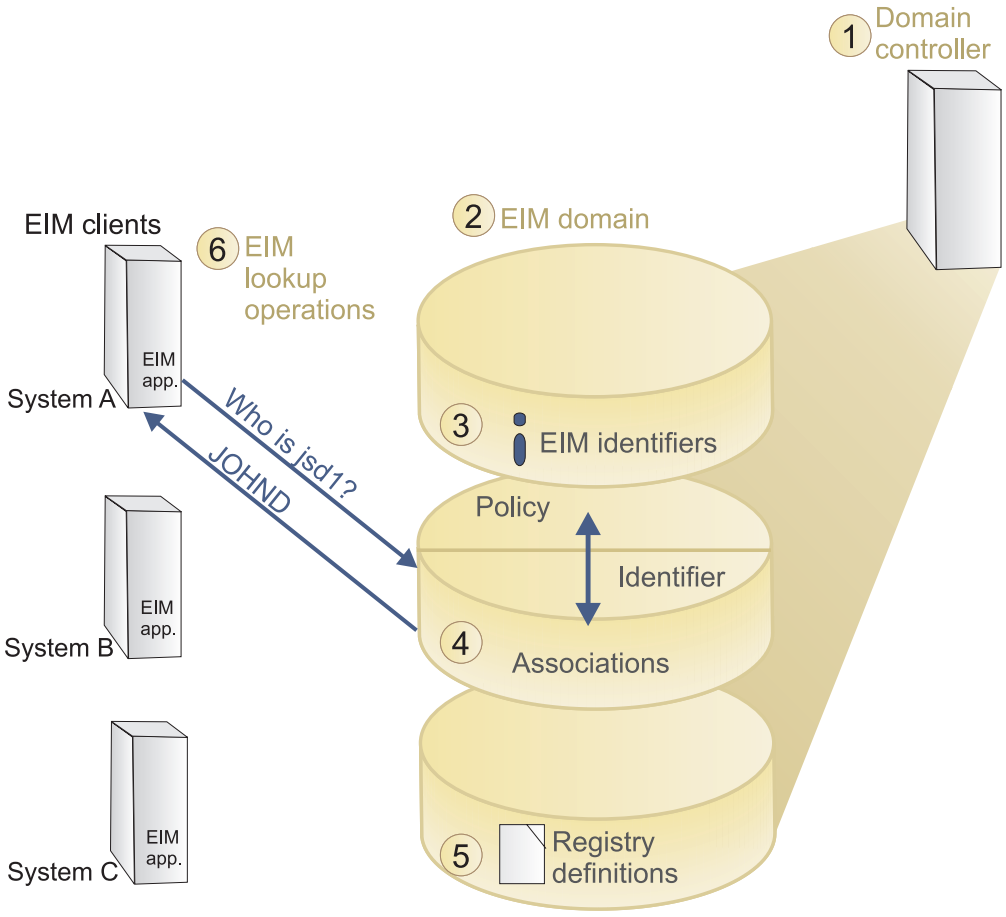


Figure 1. Overview of an EIM implementation. This shows a typical EIM implementation.

Review the following information to learn more about these EIM concepts:

- “EIM domain controller” on page 8
- “EIM domain” on page 8
- “EIM identifier” on page 10
- “EIM registry definition” on page 13
- “EIM associations” on page 17
- “EIM lookup operation” on page 25
- “EIM access control” on page 30

EIM domain controller

The EIM domain controller is a Lightweight Directory Access Protocol (LDAP) server that is configured to manage at least one EIM domain. There are two types of z/OS LDAP servers — the Integrated Security Services LDAP server (ISS LDAP server) and the IBM Tivoli Directory Server LDAP server (IBM TDS LDAP server).

An EIM domain is an LDAP directory that consists of all the EIM identifiers, EIM associations, and user registries that are defined in that domain. Systems (EIM clients) participate in the EIM domain by using the domain data for EIM lookup operations. A minimum of one EIM domain controller must exist in the enterprise.

Currently, you can configure some IBM platforms to act as an EIM domain controller. Any system that supports the EIM APIs can participate as a client in the domain. These client systems use EIM APIs to contact an EIM domain controller to perform EIM lookup operations. Refer to “EIM lookup operation” on page 25 for more information.

The location of the EIM client determines whether the EIM domain controller is a local or remote system. The domain controller is local if the EIM client is running on the same system as the domain controller. The domain controller is remote if the EIM client is running on a separate system from the domain controller.

EIM domain

An EIM domain is a directory within a Lightweight Directory Access Protocol (LDAP) server that contains EIM data for an enterprise. An EIM domain is the collection of all the EIM identifiers, EIM associations, and user registries that are defined in that domain. Systems (EIM clients) participate in the domain by using the domain data for EIM lookup operations.

An EIM domain is different from a user registry. A user registry defines a set of user identities known to and trusted by a particular instance of an operating system or application. A user registry also contains the information needed to authenticate the user of the identity. Additionally, a user registry often contains other attributes such as user preferences, system privileges, or personal information for that identity.

In contrast, an EIM domain refers to user identities that are defined in user registries. An EIM domain contains information about the relationship between identities in various user registries (user name, registry type, and registry instance) and the actual people or entities that these identities represent. Because EIM tracks relationship information only, there is nothing to synchronize between user registries and EIM.

The following figure shows the data that is stored within an EIM domain. This data includes EIM identifiers, EIM registry definitions, and EIM associations. EIM data defines the relationship between user identities and the people or entities that these identities represent in an enterprise.

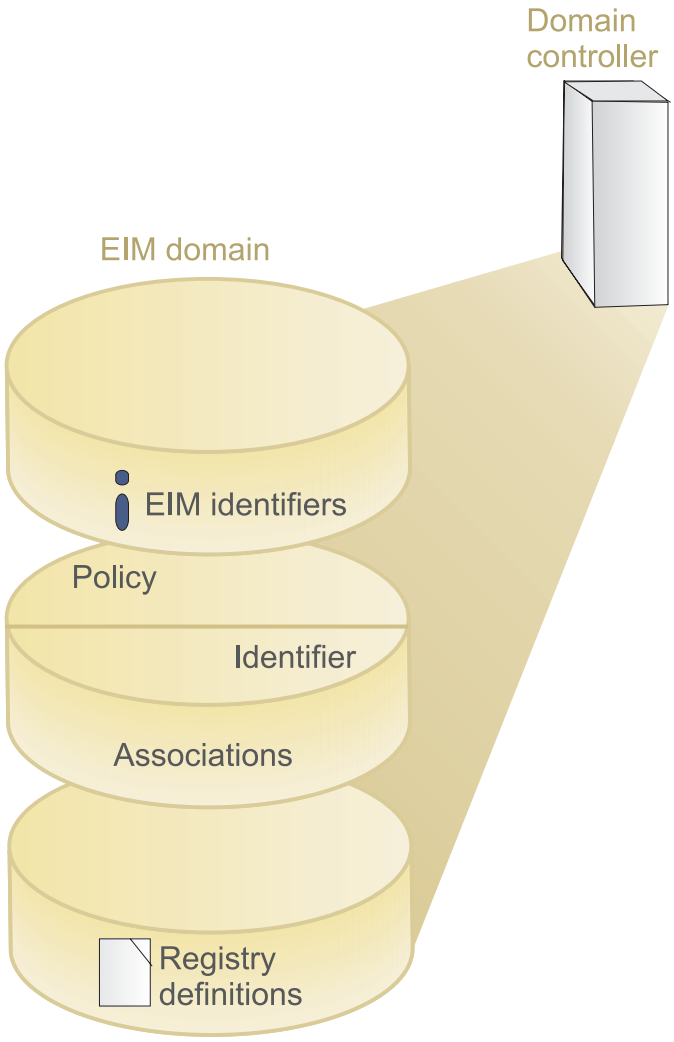


Figure 2. EIM domain and the data that is stored within the domain

EIM data includes:

EIM identifier
Each EIM identifier that you create represents a person or entity (such as a print server or a file server) within an enterprise. See “EIM identifier” on page 10 for more information about this concept.

EIM registry definition
Each EIM registry definition that you create represents an actual user registry (and the user identity information it contains) that exists on a system within the enterprise. Once you define a specific user registry in EIM, that user registry can participate in the EIM domain. You can create two types of registry definitions, one type refers to system user registries and the other type refers to application user registries. See “EIM registry definition” on page 13 for more information about this concept.

EIM association
Each EIM association that you create represents the relationship between an EIM identifier and an associated identity within an enterprise. You must define associations so that EIM clients can use EIM APIs to perform successful EIM lookup operations. These EIM lookup operations search an EIM domain for defined associations between EIM identifiers and user

identities in recognized user registries. Associations provide the information that ties an EIM identifier to a specific user identity in a specific user registry. See “EIM lookup operation” on page 25 for more information about this concept. There are two different types of associations that you can create:

- **Identifier associations**. Identifier associations allow you to define a one-to-one relationship between user identities through an EIM identifier defined for an individual. Each EIM identifier association that you create represents a single, specific relationship between an EIM identifier and an associated user identity within an enterprise. Identifier associations provide the information that ties an EIM identifier to a specific user identity in a specific user registry and allow you to create one-to-one identity mapping for a user. Identity associations are especially useful when individuals have user identities with special authorities and other privileges that you want to specifically control by creating one-to-one mappings between their user identities.
- **Policy associations.** Policy associations allow you to define a relationship between a group of user identities in one or more user registries and an individual user identity in another user registry. Each EIM policy association that you create results in a many-to-one mapping between the source group of user identities in one user registry and a single target user identity. Typically, you create policy associations to map a group of users who all require the same level of authorization to a single user identity with that level of authorization.

Once you create your EIM identifiers, registry definitions, and associations, you can begin using EIM to more easily organize and work with user identities within your enterprise.

EIM identifier

An EIM identifier represents a person or entity in an enterprise. A typical network consists of various hardware platforms and applications and their associated user registries. Most platforms and many applications use platform-specific or application-specific user registries. These user registries contain all of the user identification information for users who work with those servers or applications.

When you create an EIM identifier and associate it with the various user identities for a person or entity, it becomes easier to build heterogeneous, multiple-tier applications (for example, a single sign-on environment). When you create an EIM identifier and associations, it also becomes easier to build and use tools that simplify the administration involved with managing every user identity that a person or entity has within the enterprise.

EIM identifier representing a person

The following figure shows an example of an EIM identifier that represents a person named *John Day* and his various user identities in an enterprise. In this example, the person *John Day* has four user identities in four different user registries, which are `johnday`, `jsd1`, `JOHND`, and `JDay`.

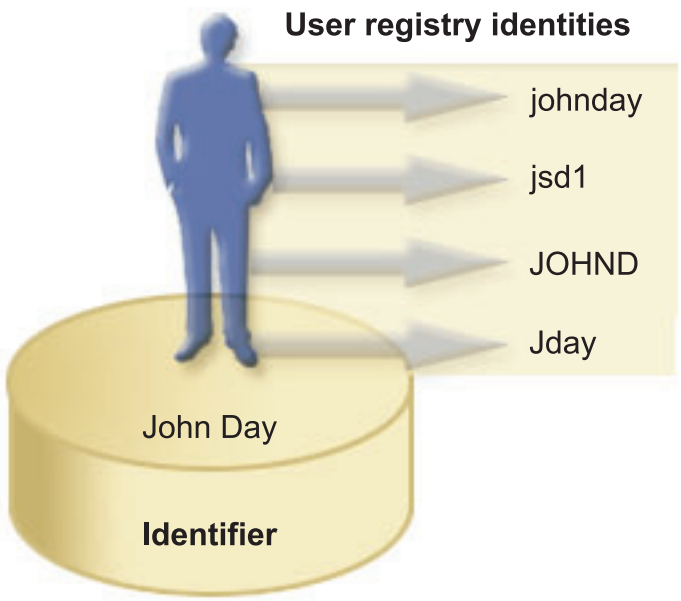


Figure 3. The relationship between the EIM identifier for a real person, John Day, and his various user identities.

In EIM, you can create associations that define the relationships between the *John Day* identifier and each of the different user identities for *John Day*. By creating these associations to define these relationships, you and others can write applications that use the EIM APIs to look up a needed, but unknown, user identity based on a known user identity.

EIM identifier representing an entity

In addition to representing users, EIM identifiers can represent entities within your enterprise as Figure 4 illustrates. For example, often the print server function in an enterprise runs on multiple systems. In the following example, there are three print servers in the enterprise running on three different systems under three different user identities of `pserverID1`, `pserverID2`, and `pserverID3`.

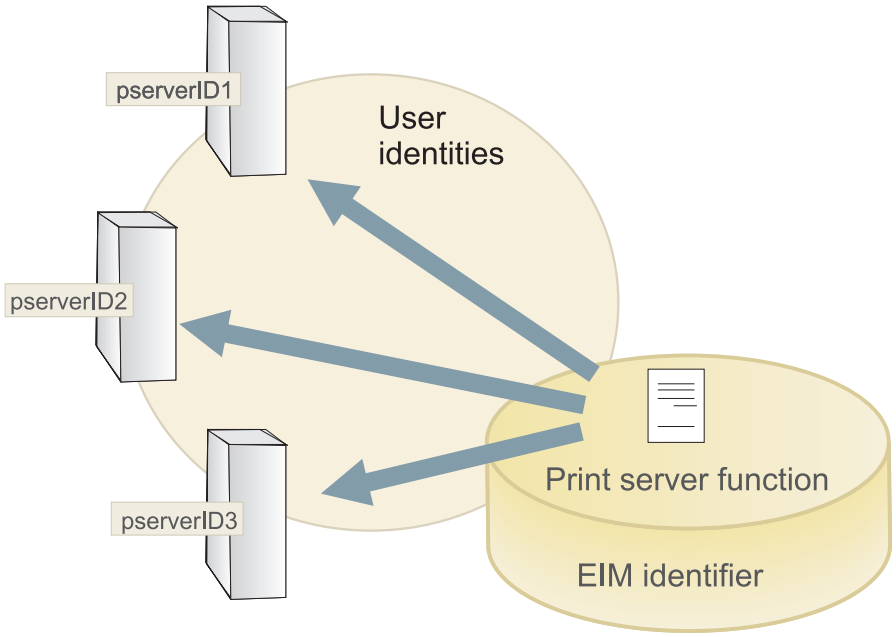


Figure 4. The relationship between the EIM identifier that represents the printer server function, and the various identities for that function.

With EIM, you can create a single identifier that represents the print server function within the entire enterprise. In this example, the EIM identifier print server function represents the actual print server function entity in the enterprise. Associations are created to define the relationships between the EIM identifier (print server function) and each of the user identities for this function (`pserverID1`, `pserverID2`, and `pserverID3`). These associations allow application developers to use EIM lookup operations to find a specific print server function. Application providers can then write distributed applications that manage the print server function more easily across the enterprise.

EIM identifiers and aliasing

You can also create aliases for EIM identifiers. Aliases can aid in locating a specific EIM identifier when performing an EIM lookup operation. For example, aliases can be useful in situations where someone's legal name is different from the name that that person is known as.

EIM identifier names must be unique within an EIM domain. Aliases can help address situations where using unique identifier names can be difficult. For example, different individuals within an enterprise can share the same name, which can be confusing if you are using proper names as EIM identifiers.

The following figure illustrates an example in which an enterprise has two users named *John S. Day*. The EIM administrator creates two different EIM identifiers to distinguish between them: `John S Day1` and `John S. Day2`. However, which real *John S. Day* is represented by each of these identifiers is not readily apparent.

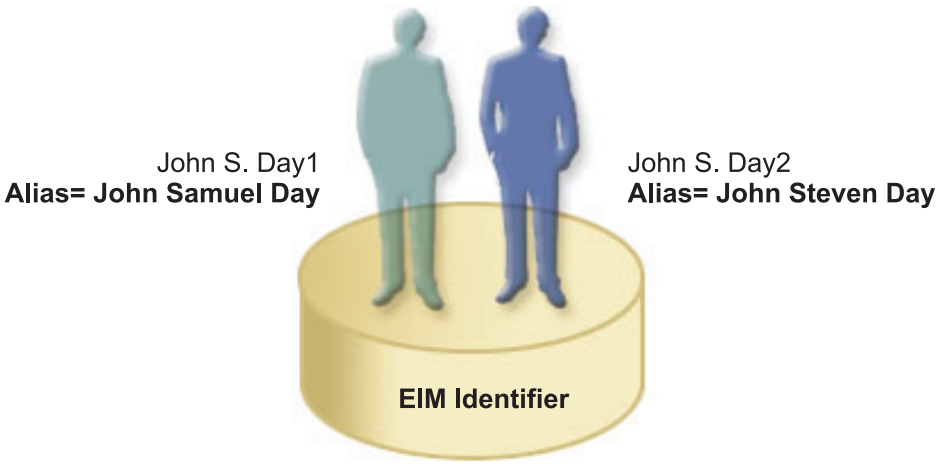


Figure 5. Aliases for the two EIM identifiers based on the shared proper name, John S. Day.

By using aliases, the EIM administrator can provide additional information about the individual for each EIM identifier. This information can also be used in an EIM lookup operation to distinguish between the users that the identifier represents. For example, the alias for `John S. Day1` might be `John Samuel Day` and the alias for `John S. Day2` might be `John Steven Day`.

Each EIM identifier can have multiple aliases to identify which *John S. Day* the EIM identifier represents. The EIM administrator might add another alias to each of the EIM identifiers for the two individuals to further distinguish between them. For example, the additional aliases might contain each user's employee number, department number, job title, or other distinguishing attribute.

You can use the alias information to aid in locating a specific EIM identifier. For example, an application that uses EIM may specify an alias that it uses to find the appropriate EIM identifier for the application. An administrator can add this alias to an EIM identifier so that the application can use the alias rather than the unique identifier name for EIM operations. An application can specify this information when using the Get EIM Target Identities from the Identifier (eimGetTargetFromIdentifier) API to perform an EIM lookup operation to find the appropriate user identity it needs.

EIM registry definition

An EIM registry definition represents an actual user registry that exists on a system within the enterprise. A user registry operates like a directory and contains a list of valid user identities for a particular system or application. A basic user registry contains user identities and their passwords. One example of a user registry is the z/OS Security Server Resource Access Control Facility (RACF) registry. User registries can contain other information as well. For example, a Lightweight Directory Access Protocol (LDAP) directory contains bind distinguished names, passwords, and access controls to data that is stored in LDAP. Other examples of common user registries are a Kerberos key distribution center (KDC) and the OS/400® user profiles registry.

You can also define user registries that exist within other user registries. Some applications use a subset of user identities within a single instance of a user registry. For example, the z/OS Security Server (RACF) registry can contain specific

user registries that are a subset of users within the overall RACF user registry. To model this behavior, EIM allows administrators to create two kinds of EIM registry definitions:

- System registry definitions
- Application registry definitions

EIM registry definitions provide information regarding those user registries in an enterprise. The administrator defines these registries to EIM by providing the following information:

- A unique, arbitrary EIM registry name
- The type of user registry

Each registry definition represents a specific instance of a user registry. Consequently, you should choose an EIM registry definition name that helps you to identify the particular instance of the user registry. For example, you could choose the TCP/IP host name for a system user registry, or the host name combined with the name of the application for an application user registry. You can use any combination of alphanumeric characters, mixed case, and spaces to create unique EIM registry definition names.

There are a number of predefined user registry types that EIM provides to cover most operating system user registries. These include:

- AIX®
- Domino - long name
- Domino - short name
- Kerberos
- Kerberos - case sensitive
- LDAP
- Linux®
- Policy director
- Novell Directory Server
- OS/400
- Tivoli Access Manager
- RACF
- Windows® - local
- Windows domain (Kerberos)
- X.509

Note: Although the predefined registry definition types cover most operating system user registries, you may need to create a registry definition for which EIM does not include a predefined registry type. You have two options in this situation. You can either use an existing registry definition which matches the characteristics of your user registry or you can define a private user registry type. For example in the following figure, the administrator followed the process required and defined the type of registry as WebSphere® Third-Party Authentication (LTPA) for the System_A_WAS application registry definition.

In the following figure, the administrator creates EIM registry definitions for user registries representing System A, System B, and System C and a Windows Active Directory that contains users' Kerberos principals with which users log into their desk top workstations. In addition, the administrator created an application registry

definition for WebSphere Lightweight Third-Party Authentication (LTPA), which runs on System A. The registry definition name that the administrator uses helps to identify the specific occurrence of the type of user registry. For example, an IP address or host name is often sufficient for many types of user registries. In this example, the administrator identifies the specific user registry instance by using `System_A_WAS` as the registry definition name to identify this specific instance of the WebSphere LTPA application. In addition to the name, the administrator also provides the type of registry as `System_A`.

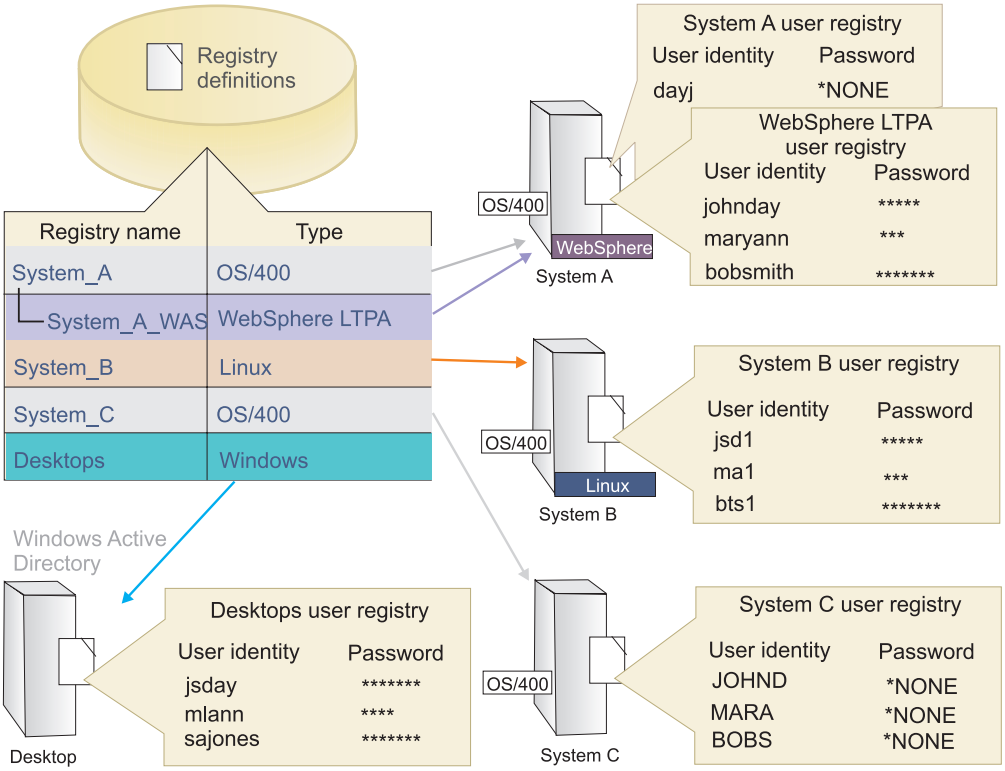


Figure 6. EIM registry definitions for five real-world user registries

You can also define user registries that exist within other user registries. For example, the z/OS Security Server (RACF) registry can contain specific user registries that are a subset of users within the overall RACF user registry. For a more detailed example of how this works, see “System and application registry definitions” on page 16.

EIM registry definitions and aliasing

You can also create aliases for EIM registry definitions. You can use predefined alias types or you can define your own alias types to use. The predefined alias types include:

- Domain Name System (DNS) host name
- Kerberos realm
- Issuer distinguished name (DN)
- Root distinguished name (DN)
- TCP/IP address
- LDAP DNS host name
- Other

An alias does not have to be in a specific format. You can enter a value of your own choosing for the type. For example, an application might specify that the administrator assign an alias with an alias type of `appl` and alias name of `source registry`. The application can then use the eimGetRegistryNameFromAlias() API and specify the alias type and name for the API to retrieve the user registry the application needs.

This alias support allows programmers to write applications without having to know in advance the arbitrary EIM registry name chosen by the administrator who deploys the application. Application documentation can provide the EIM administrator with the alias name and type that the application uses. Using this information, the EIM administrator can assign this alias name to the EIM registry definition that represents the actual user registry that the administrator wants the application to use.

When the administrator adds the alias to the EIM registry definition, the application can perform an alias lookup to find the EIM registry name at initialization. The alias lookup allows the application to determine the EIM registry name or names to use as input to the APIs that perform a lookup operation, as discussed in "EIM lookup operation" on page 25.

For example, an application that is written to use EIM may specify either a source registry alias or a target registry alias, or aliases for both. When you assign these aliases to the appropriate registry definitions, the application can perform an alias lookup to find the EIM registry definition or definitions that match the aliases in the application. This alias lookup ensures that the application uses the user registry or user registries that the administrator wants it to use. Based on application requirements, an administrator can assign multiple aliases to a single registry definition.

System and application registry definitions

Some applications use a subset of user identities within a single instance of a user registry. EIM allows administrators to model this scenario by providing two kinds of EIM registry definitions, system and application.

A system registry definition represents a distinct registry within a workstation or server. You can create a system registry definition when the registry in the enterprise has one of the following traits:

- The registry is provided by an operating system, such as AIX, OS/400, or a security management product such as z/OS Security Server Resource Access Control Facility (RACF).
- The registry contains user identities that are unique to a specific application, such as Lotus Notes.
- The registry contains distributed user identities, such as Kerberos principals or Lightweight Directory Access Protocol (LDAP) distinguished names.

An application registry definition is an entry in EIM that you create to describe and represent a subset of user identities that are defined in a system registry. These user identities share a common set of attributes or characteristics that allow them to use a particular application or set of applications. Application registry definitions represent user registries that exist within other user registries. For example, the z/OS Security Server (RACF) registry can contain specific user registries that are a subset of users within the overall RACF user registry. Because the of this relationship, you must specify the name of the parent system registry for any

application registry definition that you create. You can create an application registry definition when the user identities have the following traits:

- The user identities for the application or set of applications are not stored in a user registry specific to the application or set of applications.
- The user identities for the application or set of applications are stored in a system registry that contains user identities for other applications.

EIM lookup operations perform correctly regardless of whether an EIM administrator defines a registry either as system or application. However, separate registry definitions allow mapping data to be managed on an application basis. The responsibility of managing application-specific mappings can be assigned to an administrator for a specific registry.

For example, the following figure shows how an EIM administrator created a system registry definition to represent a z/OS Security Server RACF registry. The administrator also created an application registry definition to represent the user identities within the RACF registry that use z/OS UNIX System Services (z/OS UNIX). System C contains a RACF user registry that contains information for three user identities, DAY1, ANN1, and SMITH1. Two of these user identities (DAY1 and SMITH1) access z/OS UNIX on System C. These user identities are actually RACF users with unique attributes that identify them as z/OS UNIX users. Within the EIM registry definitions, the EIM administrator defined System_C_RACF to represent the overall RACF user registry. The administrator also defined System_C_UNIX to represent the user identities that have z/OS UNIX attributes.

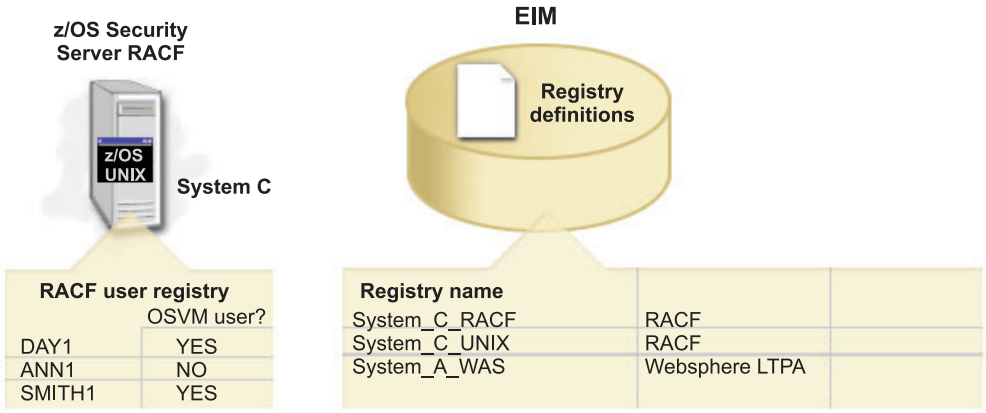


Figure 7. EIM registry definitions for both the RACF user registry and for a subset of the RACF registry

EIM associations

An EIM association is an entry that you create in an EIM domain to define a relationship between user identities in different user registries. The type of association that you create determines whether the defined relationship is direct or indirect. You can create one of two types of associations in EIM: identifier associations and policy associations. You can use policy associations instead of, or in combination with, identifier associations. How you use associations depends on your overall EIM implementation plan.

Identifier associations

An EIM identifier represents a specific person or entity in the enterprise. An EIM identifier association describes a relationship between an EIM identifier and a single user identity in a user registry that also represents that person. When you create associations between an EIM identifier and all of a person's or entity's user identities, you provide a single, complete understanding of how that person or entity uses the resources in an enterprise

User identities can be used for authentication, authorization, or both. Authentication is the process of verifying that an entity or person who provides a user identity has the right to assume that identity. Verification is often accomplished by forcing the person who submits the user identity to provide secret or private information associated with the user identity, such as a password. Authorization is the process of ensuring that a properly authenticated user identity can only perform functions or access resources for which the identity has been given privileges. In the past, nearly all applications were forced to use the identities in a single user registry for both authentication and authorization. By using EIM lookup operations, applications now can use the identities in one user registry for authentication while they use associated user identities in a different user registry for authorization. Refer to "EIM lookup operation" on page 25 for more information.

An EIM association is a relationship between an EIM identifier that represents a specific person and a single user identity in a user registry that also represents that person. When you create associations between an EIM identifier and all of a person's or entity's user identities, you provide a single, complete understanding of how that person or entity uses the resources in an enterprise. EIM provides APIs that allow applications to find an unknown user identity in a specific (target) user registry by providing a known user identity in some other (source) user registry. This process is called identity mapping.

In EIM, an administrator can define three different types of associations to describe the relationship between an EIM identifier and a user identity. Identifier associations can be any of the following types: source, target, or administrative. The type of association that you create is based on how the user identity is used. For example, you create source and target associations for those user identities that you want to participate in mapping lookup operations. Typically, if a user identity is used for authentication, you create a source association for it. You then create target associations for those user identities that are used for authorization.

Before you can create an association, you first must create the appropriate EIM identifier and the appropriate EIM registry definition for the user registry that contains the associated user identity. An association defines a relationship between an EIM identifier and a user identity by using the following information:

- EIM identifier name
- User identity name
- EIM registry definition name
- Association type
- Lookup information to further identity the target user identity in a target association (optional).

An administrator can create different types of associations between an EIM identifier and a user identity based on how the user identity is used. User identities can be used for authentication, authorization, or both.

Source and target association relationship

In EIM, there are three types of associations that an administrator can define between an EIM identifier and a user identity. These types are source, target, and administrative associations.

Source association

When a user identity is used for authentication, that user identity should have a source association with an EIM identifier. A source association allows the user identity to be used as the source in an EIM lookup operation to find a different user identity that is associated with the same EIM identifier. If a user identity with only a source association is used as the target identity in an EIM lookup operation, no associated user identities are returned. To ensure successful mapping lookup operations for EIM identifiers, source and target associations must be used together for a single EIM identifier.

Target association

When a user identity is used for authorization rather than for authentication, that user identity should have a target association with an EIM identifier. A target association allows the user identity to be returned as the result of an EIM lookup operation. If a user identity with only a target association is used as the source identity in an EIM lookup operation, no associated user identities are returned.

It might be necessary to create both a target and a source association for a single user identity. This is required when an individual uses a single system as both a client and a server or for individuals who act as administrators. For example, a user normally authenticates to a Windows platform and runs applications that access an AIX server. Because of the user's job responsibilities, the user must occasionally also log directly into an AIX server. In this situation you would create both source and target associations between the AIX user identity and the person's EIM identifier. User identities that represent end users normally need a target association only.

To ensure successful mapping lookup operations, you need to create at least one source and one or more target associations for a single EIM identifier. Typically, you create a target association for each user identity in a user registry that the person can use for authorization to the system or application to which the user registry corresponds.

The following figure shows an example of a source and a target association. In this example, the administrator created two associations for the EIM identifier `John Day` to define the relationship between this identifier and two associated user identities. The administrator created a source association for johnday, the WebSphere Lightweight Third-Party Authentication (LTPA) user identity in the `System_A_WAS` user registry. The administrator also created a target association for `jsd1`, the OS/400 user profile in the System B user registry. These associations provide a means for applications to obtain an unknown user identity (the target, `jsd1`) based on a known user identity (the source, `johnday`) as part of an EIM lookup operation.

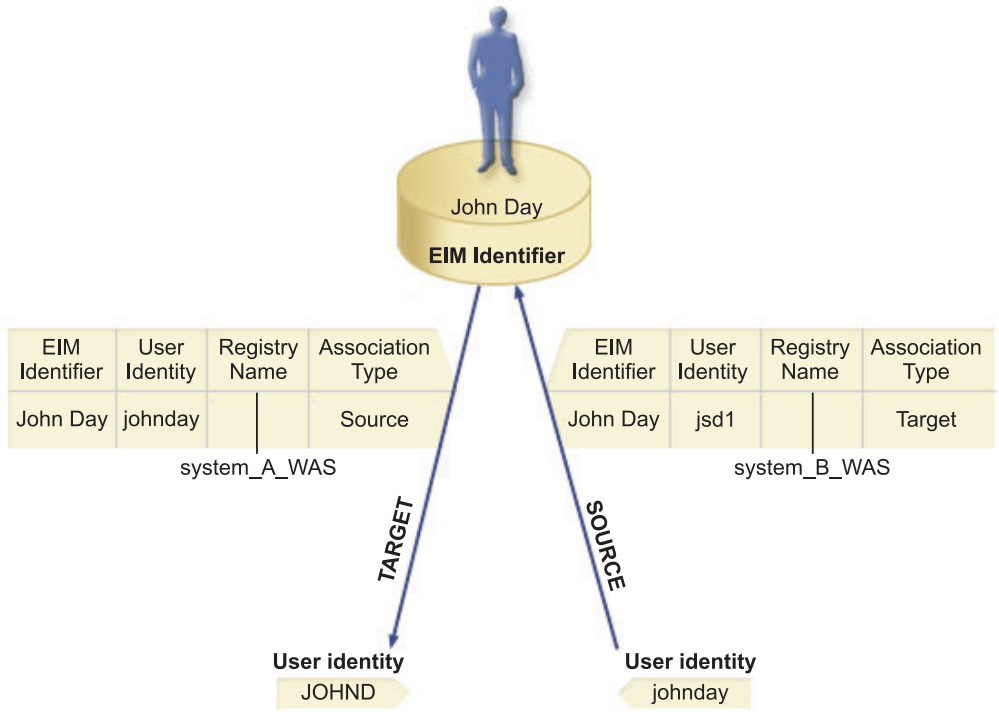


Figure 8. EIM target and source associations for the EIM identifier John Day

Administrative association

An administrative association for an EIM identifier is typically used to show that the person or entity represented by the EIM identifier owns a user identity that requires special considerations for a specified system. This type of association can be used, for example, with highly sensitive user registries.

Due to the nature of what an administrative association represents, an EIM lookup operation that supplies a source user identity with an administrative association returns no results. Similarly, a user identity with an administrative association is never returned as the result of an EIM lookup operation.

The following figure shows an example of an administrative association. In this example, John Day has one user identity on System A and another user identity on System B, which is a highly secure system. The system administrator wants to ensure that users authenticate to System B by using only the local user registry of this system. The administrator does not want to allow an application to authenticate John Day to the system by using some foreign authentication mechanism. By using an administrative association for the JDay user identity on System B, the EIM administrator can see that John Day owns an account on System B, but EIM does not return information about the JDay identity in EIM lookup operations. Even if applications exist on this system that use EIM lookup operations, they cannot find user identities that have administrative associations.

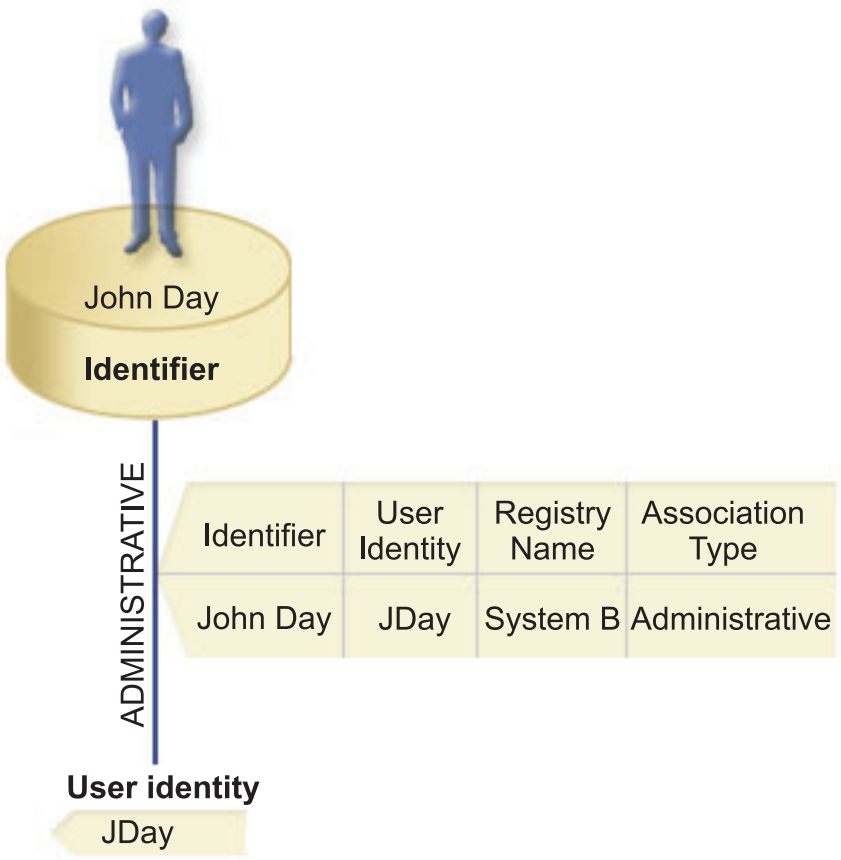


Figure 9. EIM administrative associations for the EIM identifier, John Day

Policy associations

EIM's mapping policy support allows an EIM administrator to create and use policy associations to define a relationship between multiple user identities in one or more user registries and a single user identity in another user registry. Policy associations use EIM mapping policy support to create many-to-one mappings between user identities without involving an EIM identifier. You can use policy associations instead of, or in combination with, identifier associations that provide one-to-one mappings between an EIM identifier and a single user identity.

A policy association affects only those user identities for which specific individual EIM associations do not exist. When specific identifier associations exist between an EIM identifier and user identities, then the target user identity from the identifier association is returned to the application performing the lookup operation, even when a policy association exists and the use of policy associations is enabled.

With EIM, you can create three different types of policy associations: default domain, default registry, and certificate filter policy associations. Because you can use policy associations in a variety of overlapping ways, you should have a thorough understanding of EIM mapping policy support and how lookup operations work before you create and use policy associations.

Default domain policy associations

A default domain policy association is one type of policy association that you can use to create many-to-one mappings between user identities. You can use a default domain policy association to map a source set of multiple

user identities (in this case, all users in the domain) to a single target user identity in a specified target user registry. In a default domain policy association, all users in the domain are the source of the policy association and are mapped to a single target registry and target user identity.

To use a default domain policy association, you must enable mapping lookups using policy associations for the domain. You must also enable mapping lookups for the target user registry of the policy association. When you configure this enablement, the user registries in the policy association can participate in mapping lookup operations.

The default domain policy association takes effect when a mapping lookup operation is not satisfied by identifier associations, certificate filter policy associations, or default registry policy associations for the target registry. The result is that all user identities in the domain are mapped to the single target user identity as specified by the default domain policy association. For example, you create a default domain policy association with a target user identity of `John_Day` in target registry `Registry_xyz` and you have not created any identifier associations or other policy associations that map to this user identity. Therefore, when `Registry_xyz` is specified as the target registry in lookup operations, the default domain policy ensures that the target user identity of `John_Day` is returned for all user identities in the domain that do not have any other associations defined for them.

You specify the following to define a default domain policy association:

- **Target registry**. The name of an EIM registry definition which contains the user identity to which all user identities in the domain are to be mapped.
- **Target user**. The name of the user identity that is returned as the target of an EIM mapping lookup operation based on this policy association.

You can define a default domain policy association for each registry in the domain. If two or more domain policy associations refer to the same target registry, you must define unique lookup information for each of these policy associations to ensure that mapping lookup operations can distinguish among them. Otherwise, mapping lookup operations may return multiple target user identities. As a result of these ambiguous results, applications that rely on EIM may not be able to determine the exact target user identity to use.

Default registry policy associations

A default registry policy association is one type of policy association that you can use to create many-to-one mappings between user identities. You can use a default registry policy association to map a source set of multiple user identities (in this case those in a single registry) to a single target user identity in a specified target user registry. In a default registry policy association, all users in a single registry are the source of the policy association and are mapped to a single target registry and target user.

To use default registry policy associations, you must enable mapping lookups using policy associations for the domain. You must also enable mapping lookups for the source registry and enable mapping lookups and the use of policy associations for the target user registry of the policy association. When you configure this enablement, the user registries in the policy association can participate in mapping lookup operations.

The default registry policy association takes effect when a mapping lookup operation is not satisfied by identifier associations, certificate filter policy associations, or other default registry policy associations for the target

registry. The result is that all user identities in the source registry are mapped to the single target user identity as specified by the default registry policy association.

For example, you create a default registry policy association that has a source registry of `my_realm.com`, which are principals in a specific Kerberos realm. For this policy association, you also specify a target user identity of `general_user1` in target registry `os/400_system_reg`, which is a specific user profile in an OS/400 user registry. In this case, you have not created any identifier associations or policy associations that apply to any of the user identities in the source registry. Therefore, when `os/400_system_reg` is specified as the target registry and `my_realm.com` is specified as the source registry in lookup operations, the default registry policy association ensures that the target user identity of `general_user1` is returned for all user identities in `my_realm.com` that do not have any specific identifier associations or certificate filter policy associations defined for them.

You specify the following to define a default registry policy association:

- **Source registry**. This is the registry definition that you want the policy association to use as the source of the mapping. All the user identities in this source user registry are to be mapped to the specified target user of the policy association.
- **Target registry**. The name of an EIM registry definition. The target registry must contain the target user identity to which all user identities in the source registry are to be mapped.
- **Target user**. The name of user identity that is returned as the target of an EIM mapping lookup operation based on this policy association.

You can define more than one default registry policy association. If two or more policy associations with the same source registry refer to the same target registry, you must define unique lookup information for each of these policy associations to ensure that mapping lookup operations can distinguish among them. Otherwise, mapping lookup operations may return multiple target user identities. As a result of these ambiguous results, applications that rely on EIM may not be able to determine the exact target identity to use.

Certificate filter policy associations

A type of policy association that you can use to create many-to-one mappings between user identities. You can use a certificate filter policy association to map a source set of certificates to a single target user identity in a specified target user registry.

In a certificate filter policy association, you specify a set of certificates in a single X.509 registry as the source of the policy association. These certificates are mapped to a single target registry and target user that you specify. Unlike a default registry policy association in which all users in a single registry are the source of the policy association, the scope of a certificate filter policy association is more flexible. You can specify a subset of certificates in the registry as the source. The certificate filter that you specify for the policy association is what determines its scope.

Note: When you want to map all the certificates in an X.509 user registry to a single target user identity, create and use a default registry policy association.

To use certificate filter policy associations, you must enable mapping lookups using policy associations for the domain. You must also enable

mapping lookups for the source registry and enable mapping lookups and the use of policy associations for the target user registry of the policy association. When you configure this enablement, the user registries in the policy association can participate in mapping lookup operations.

When a digital certificate is the source user identity in an EIM mapping lookup operation (after the requesting application uses eimFormatUserIdentity to format the user identity name), EIM first checks to see if there is an identifier association between an EIM identifier and the specified user identity. If none exist, EIM then compares the DN information in the certificate against the DN or partial DN information specified in the filter for the policy association. If the DN information in the certificate satisfies the criteria of the filter, EIM returns the target user identity that the policy association specified. The result is that certificates in the source X.509 registry that satisfy the certificate filter criteria are mapped to the single target user identity as specified by the certificate filter policy association.

You specify the following information to define a certificate filter policy association:

- **Source registry**. The source registry definition that you specify must be an X.509 type user registry. The certificate filter policy creates an association between user identities in this X.509 user registry and a single, specific target user identity. The association is applied to only those user identities in the registry that meet the criteria of the certificate filter that you specify for this policy.
- **Certificate filter**. Defines a set of similar user certificate attributes. The certificate filter policy association maps any certificates with these defined attributes in the X.509 user registry to a specific target user identity. You specify the filter based on a combination of the Subject distinguished name (SDN) and the Issuer distinguished name (IDN) that matches the certificates that you want to use as the source of the mapping. The certificate filter that you specify for the policy must already exist in the EIM domain.
- **Target registry**. The target registry definition that you specify is the user registry that contains the user identity to which you want to map the certificates that match the certificate filter.
- **Target user**. The target user is the name of the user identity that is returned as the target of an EIM mapping lookup operation based on this policy association.

Lookup information

You can provide optional data called lookup information to further identify a target user identity. This target user identity can be specified either in an identifier association or in a policy association. Lookup information is a unique character string that either the eimGetTargetFromSource API or the eimGetTargetFromIdentifier API can use during a mapping lookup operation to further refine the search for the target user identity that is the object of the operation. Data that you specify for lookup information corresponds to the registry users additional information parameter for these APIs.

Lookup information is necessary only when a mapping lookup operation can return more than one target user identity. A mapping lookup operation can return multiple target user identities when one or more of the following situations exist:

- An EIM identifier has multiple individual target associations to the same target registry.
- More than one EIM identifier has the same user identity specified in a source association and each of these EIM identifiers has a target association to the same target registry, although the user identity specified for each target association may be different.
- More than one default domain policy association specifies the same target registry.
- More than one default registry policy association specifies the same source registry and the same target registry.
- More than one certificate filter policy association specifies the same source X.509 registry, certificate filter, and target registry.

Note: A mapping lookup operation that returns more than one target user identity can create problems for EIM-enabled applications, including OS/400 applications and products, that are not designed to handle these ambiguous results. Consequently, you might consider redefining associations for the domain to ensure that a mapping lookup operation can return a single target user identity to ensure that base OS/400 applications can successfully perform lookup operations and map identities.

You can use lookup information to avoid situations where it is possible for mapping lookup operations to return more than one target user identity. To prevent mapping lookup operations from returning multiple target user identities, you must define unique lookup information for each target user identity in each association. This lookup information must be provided to the mapping lookup operation to ensure that the operation can return a unique target user identity. Otherwise, applications that rely on EIM may not be able to determine the exact target identity to use.

For example, you have an EIM identifier named `John Day` who has two user profiles on System A. One of these user profiles is `JDUSER` on System A and another is `JDSECADM`, which has security administrator special authority. There are two target association for the `John Day` identifier. One of these target associations is for the `JDUSER` user identity in the target registry of System_A and has lookup information of user authority specified for `JDUSER`. The other target association is for the `JDSECADM` user identity in the target registry of System_A and has lookup information of security officer specified for `JDSECADM`.

If a mapping lookup operation does not specify any lookup information, the lookup operation returns both the `JDUSER` and the `JDSECADM` user identities. If a mapping lookup operation specifies lookup information of user authority, the lookup operation returns the `JDUSER` user identity only. If a mapping lookup operation specifies lookup information of security officer, the lookup operation returns the `JDSECADM` user identity only.

If you delete the last target association for a user identity (whether it is an identifier association or a policy association), the target user identity and all lookup information is deleted from the domain as well.

EIM lookup operation

An application or an operating system uses an EIM API to perform a lookup operation so that the application or operating system can map from one user identity in one registry to another user identity in another registry. An EIM lookup operation is a process through which an application or operating system finds an

unknown associated user identity in a specific target registry by supplying some known and trusted information. Applications that use EIM APIs can perform these EIM lookup operations on information only if that information is stored in the EIM domain. An application can perform one of two types of EIM lookup operations based on the type of information the application supplies as the source of the EIM lookup operation: a user identity or an EIM identifier.

When applications or operating systems use the eimGetTargetFromSource API to obtain a target user identity for a given target registry, they must supply a user identity as the source of the lookup operation. To be used as the source in a EIM lookup operation, a user identity must have either an identifier source association defined for it or be covered by a policy association. When an application or operating system uses this API, the application or operating system must supply these pieces of information:

- A user identity as the source, or starting point of the operation.
- The EIM registry definition name for the source user identity.
- The EIM registry definition name that is the target of the EIM lookup operation. This registry definition describes the user registry that contains the user identity that the application is seeking.

When applications or operating systems use the eimGetTargetFromIdentifier API to obtain a user identity for a given target registry, they must supply an EIM identifier as the source of the EIM lookup operation. When an application uses this API, the application must supply the following pieces of information:

- A user identity as the source, or starting point of the operation.
- The EIM registry definition name that is the target of the EIM lookup operation. This registry definition describes the user registry that contains the user identity that the application is seeking.

For a user identity to be returned as the target of either type of EIM lookup operation, the user identity must have a target association defined for it. This target association can be in the form of an identifier association or a policy association.

The supplied information is passed to EIM and the lookup operation searches for and returns any target user identities, by searching EIM data in the following order:

1. Identifier target association for an EIM identifier. The EIM identifier is identified in one of two ways: It is supplied by the eimGetTargetFromIdentifier API. Or, the EIM identifier is determined from information supplied by the eimGetTargetFromSource API.
2. Certificate filter policy association.
3. Default registry policy association.
4. Default domain policy association.

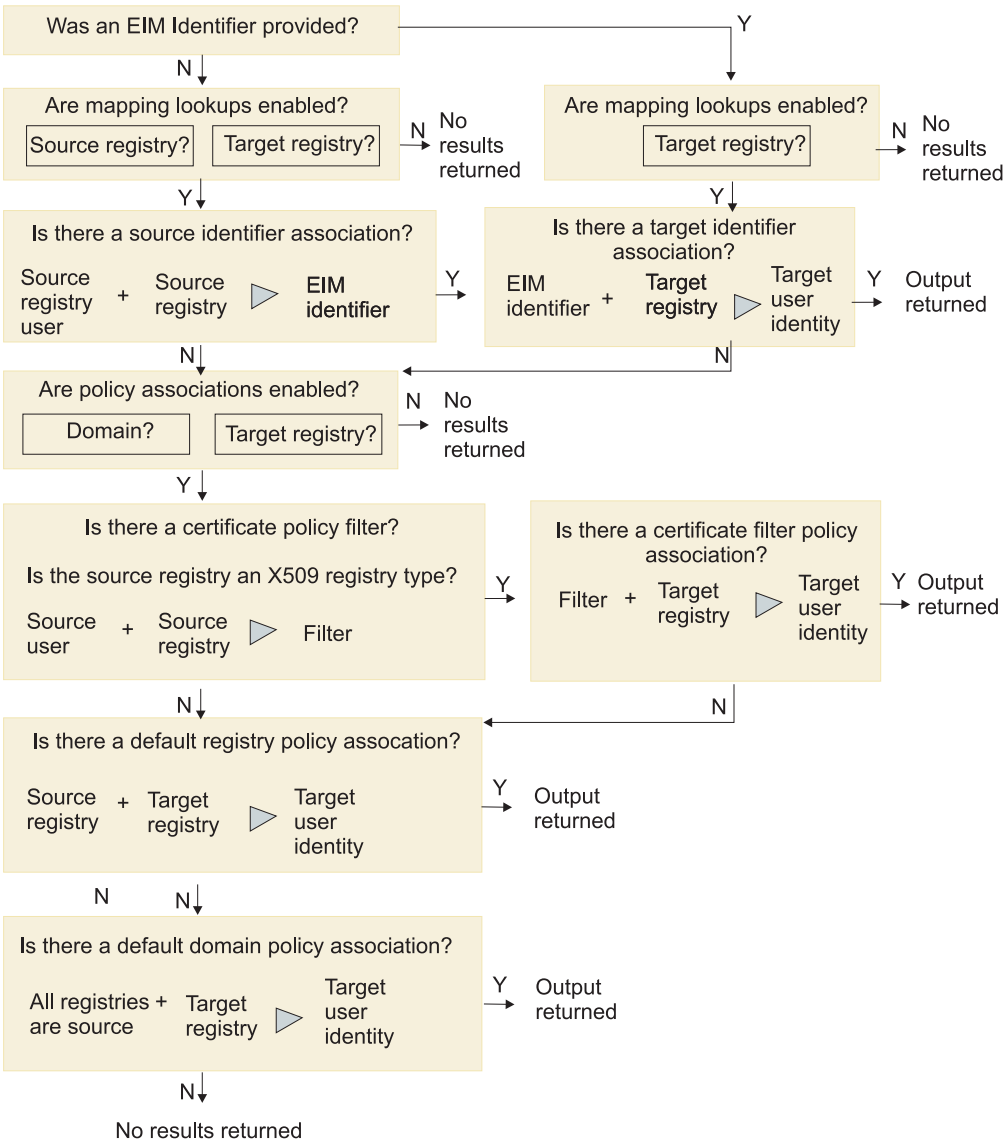


Figure 10. EIM lookup operation general processing flow chart

The lookup operation search flows in this manner:

1. The lookup operation checks whether mapping lookups are enabled. The lookup operation determines whether mapping lookups are enabled for the specified source registry, the specified target registry, or both specified registries. If mapping lookups are not enabled for one or both of the registries, then the lookup operation ends without returning a target user identity
2. The lookup operation checks whether there are identifier associations that match the lookup criteria. If an EIM identifier was provided, the lookup operation uses the specified EIM identifier name. Otherwise, the lookup operation checks whether there is a specific identifier source association that matches the supplied source user identity and source registry. If there is one, the lookup operation uses it to determine the appropriate EIM identifier name. The lookup operation then uses the EIM identifier name to search for an identifier target association for the EIM identifier that matches the specified target EIM registry definition name. If there is an identifier target association that matches, the lookup operation returns the target user identity defined in the target association

3. The lookup operation checks whether the use of policy associations are enabled. The lookup operation checks whether the domain is enabled to allow mapping lookups using policy associations. The lookup operation also checks whether the target registry is enabled to use policy associations. If the domain is not enabled for policy associations or the registry is not enabled for policy associations, then the lookup operation ends without returning a target user identity.
4. The lookup operation checks for certificate filter policy associations. The lookup operation checks whether the source registry is an X.509 registry type. If it is an X.509 registry type, the lookup operation checks whether there is a certificate filter policy association that matches the source and target registry definition names. The lookup operation checks whether there are certificates in the source X.509 registry that satisfy the criteria specified in the certificate filter policy association. If there is a matching policy association and there are certificates that satisfy the certificate filter criteria, the lookup operation returns the appropriate target user identity for that policy association.
5. The lookup operation checks for default registry policy associations. The lookup operation checks whether there is a default registry policy association that matches the source and target registry definition names. If there is a matching policy association, the lookup operation returns the appropriate target user identity for that policy association.
6. The lookup operation checks for default domain policy associations. The lookup operation checks whether there is a default domain policy association defined for the target registry definition. If there is a matching policy association, the lookup operation returns the associated target user identity for that policy association.
7. The lookup operation is unable to return any results.

When an application supplies a user identity as the source, the application also must supply the EIM registry definition name for the source user identity and the EIM registry definition name that is the target of the EIM lookup operation. To be used as the source in a EIM lookup operation, a user identity must have a source association defined for it. Refer to “EIM associations” on page 17 for more information.

When an application supplies an EIM identifier as the source of the EIM lookup operation, the application must also supply the EIM registry definition name that is the target of the EIM lookup operation. For a user identity to be returned as the target of either type of EIM lookup operation, the user identity must have a target association defined for it.

The supplied information is passed to the EIM domain controller where all EIM information is stored and the EIM lookup operation searches for the source association that matches the supplied information. Based on the EIM identifier (supplied to the API or determined from the source association information), the EIM lookup operation then searches for a target association for that identifier that matches the target EIM registry definition name.

In Figure 10, the user identity `johnday` authenticates to the Websphere Application Server by using Lightweight Third-Party Authentication (LPTA) on System A. The Websphere Application Server on System A calls a native program on System B to access data on System B. The native program uses an EIM API to perform an EIM lookup operation based on the user identity on System A as the source of the operation. The application supplies the following information to perform the operation: `johnday` as the source user identity, `System_A_WAS` as the source EIM

registry definition name, and `System_B` as the target EIM registry definition name. This source information is passed to the EIM domain controller and the EIM lookup operation finds a source association that matches the information. Using the EIM identifier name, the EIM lookup operation searches for a target association for the `John Day` identifier that matches the target EIM registry definition name for `System_B`. When the matching target association is found, the EIM lookup operation returns the jsd1 user identity to the application.

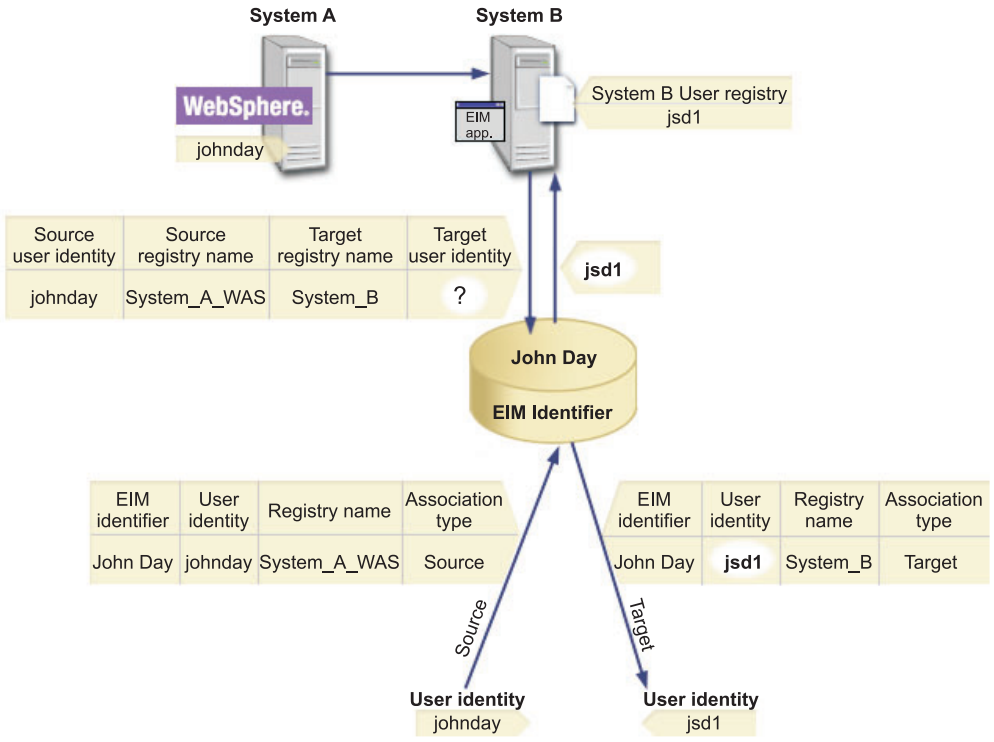


Figure 11. EIM lookup operation based on the known user identity johnday

Mapping policy support and enablement

EIM mapping policy support allows you to use policy associations as well as specific identifier associations in an EIM domain. You can use policy associations instead of, or in combination with, identifier associations.

EIM mapping policy support provides a means of enabling and disabling the use of policy associations for the entire domain, as well as for each specific target user registry. EIM also allows you to set whether a specific registry can participate in mapping lookup operations in general. Consequently, you can use mapping policy support to more precisely control how mapping lookup operations return results.

The default setting for an EIM domain is that mapping lookups that use policy associations are disabled for the domain. When the use of policy associations is disabled for the domain, all mapping lookup operations for the domain return results only by using specific, identifier associations between user identities and EIM identifiers.

The default setting for each individual registry is that mapping lookup participation is enabled and the use of policy associations is disabled. When you enable the use of policy associations for an individual target registry, you must also ensure that this setting is enabled for the domain.

You can configure mapping lookup participation and the use of policy associations for each registry in one of the following ways:

- Mapping lookup operations can not be used for the specified registry at all. In other words, an application that performs a mapping lookup operation involving that registry will fail to return results.
- Mapping lookup operations can use specific identifier associations between user identities and EIM identifiers only. Mapping lookups are enabled for the registry, but the use of policy associations is disabled for the registry.
- Mapping lookup operations can use specific identifier associations when they exist and policy associations when specific identifier associations do not exist (all settings are enabled).

EIM access control

An EIM user is a user who possesses EIM access control based on their membership in a predefined Lightweight Directory Access Protocol (LDAP) user group for a specific domain. Specifying EIM access control for a user adds that user to a specific LDAP user group for a particular domain. Each LDAP group has authority to perform specific EIM administrative tasks for that domain. Which and what type of administrative tasks, including lookup operations, an EIM user can perform is determined by the access control group to which the EIM user belongs.

EIM access controls allow a user to perform specific administrative tasks or EIM lookup operations. Only users with EIM administrator access are allowed to grant or revoke authorities for other users. EIM access controls are granted only to user identities that are known to the EIM domain controller.

The following are brief descriptions of the functions that each EIM access control group can perform:

Lightweight Directory Access Protocol (LDAP) administrator

This access control allows the user to configure a new EIM domain. A user with this access control can perform the following functions:

- Create a domain
- Delete a domain
- Create and remove EIM identifiers
- Create and remove EIM registry definitions
- Create and remove source, target, and administrative associations
- Perform EIM lookup operations
- Retrieve associations, EIM identifiers, and EIM registry definitions
- Add, remove, and list EIM authority information

EIM administrator

This access control allows the user to manage all of the EIM data within this EIM domain. A user with this access control can perform the following functions:

- Delete a domain
- Create and remove EIM identifiers

- Create and remove EIM registry definitions
- Create and remove source, target, and administrative associations
- Perform EIM lookup operations
- Retrieve associations, EIM identifiers, and EIM registry definitions
- Add, remove, and list EIM authority information

EIM identifiers administrator

This access control allows the user to add and change EIM identifiers and manage source and administrative associations. A user with this access control can perform the following functions:

- Create an EIM identifier
- Add and remove source associations
- Add and remove administrative associations
- Perform EIM lookup operations
- Retrieve associations, EIM identifiers, and EIM registry definitions

EIM mapping lookup

This access control allows the user to conduct EIM lookup operations. A user with this access control can perform the following functions:

- Perform EIM lookup operations
- Retrieve associations, EIM identifiers, and EIM registry definitions

EIM registries administrator

This access control allows the user to manage all EIM registry definitions. A user with this access control can perform the following functions:

- Add and remove target associations
- Perform EIM lookup operations
- Retrieve associations, EIM identifiers, and EIM registry definitions

EIM registry X administrator

This access control allows the user to manage a specific EIM registry definition. Membership in this access control group also allows the user to add and remove target associations only for a specified user registry definition. To take full advantage of mapping lookup operations and policy associations, a user with this access control should also have EIM mapping operations access control. This access control allows a user to:

- Create, remove, and list target associations for the specified EIM registry definitions only
- Add and remove default domain policy associations
- Add and remove policy associations for the specified registry definitions only
- Add certificate filters for the specified registry definitions only
- Enable and disable mapping lookups for the specified registry definitions only
- Add and remove policy associations only for the specified registries
- Retrieve EIM identifiers
- Retrieve identifier associations and certificate filters for the specified registry definitions only
- Add and remove target associations for the specific EIM registry definition
- Perform EIM lookup operations

- Retrieve EIM registry definition information for the specified registry definitions only

Each of the following tables is organized by the EIM task that the API performs. Each table displays each EIM API, the different EIM access controls, and the access each of these access controls has to certain EIM functions. Keep in mind there are also Java versions of these APIs, in case you need to work with z/OS applications written in Java. For more information on Java APIs, see “Java APIs” on page 159.

Table 1. Working with domains

EIM API	LDAP admin	EIM admin	Identifier admin	Identity mapping operations	Registry admin	Admin for selected registries
eimChangeDomain	X	X				
eimCreateDomain	X					
eimDeleteDomain	X	X				
eimListDomains	X	X				

Table 2. Working with identifiers

EIM API	LDAP admin	EIM admin	Identifier admin	Identity mapping operations	Registry admin	Admin for selected registries
eimAddIdentifier	X	X	X			
eimChangeIdentifier	X	X	X			
eimListIdentifiers	X	X	X	X	X	X
eimRemoveIdentifier	X	X				
eimGetAssociated Identifiers	X	X	X	X	X	X

Table 3. Working with registries

EIM API	LDAP admin	EIM admin	Identifier admin	Identity mapping operations	Registry admin	Admin for selected registries
eimAddApplicationRegistry	X	X				
eimAddSystemRegistry	X	X				
eimChangeRegistry	X	X			X	X
eimChangeRegistryUser	X	X			X	X
eimChangeRegistryAlias	X	X			X	X
eimGetRegistry NameFromAlias	X	X	X	X	X	X
eimListRegistries	X	X	X	X	X	X
eimListRegistryAliases	X	X	X	X	X	X
eimListRegistry Associations	X	X	X	X	X	X
eimListRegistyUsers	X	X	X	X	X	X
eimRemoveRegistry	X	X				

For eimAddAssociation() and eimRemoveAssociation() APIs there are four parameters that determine the type of association that is either being added or removed. The authority to these APIs differs based on the type of association specified in these parameters. In the following table, the type of association is included for each of these APIs.

Table 4. Working with associations

EIM API	LDAP admin	EIM admin	Identifier admin	Identity mapping operations	Registry admin	Admin for selected registries
eimAddAssociation (admin)	X	X	X			

Table 4. Working with associations (continued)

eimAddAssociation (source)	X	X	X			
eimAddAssociation (source and target)	X	X	X		X	X
eimAddAssociation (target)	X	X			X	X
eimListAssociations	X	X	X	X	X	X
eimRemoveAssociation (admin)	X	X	X			
eimRemoveAssociation (source)	X	X	X			
eimRemoveAssociation (source and target)	X	X	X		X	X
eimRemoveAssociation (target)	X	X	X		X	X

Table 5. Working with mappings

EIM API	LDAP admin	EIM admin	Identifier admin	Identity mapping operations	Registry admin	Admin for selected registries
eimGetAssociated Identifiers	X	X	X	X	X	X
eimGetRegistryNameFrom Alias	X	X	X	X	X	X
eimGetTargetFromIdentifier	X	X	X	X	X	X
eimGetTargetFromSource	X	X	X	X	X	X

Table 6. Working with policy associations

EIM API	LDAP admin	EIM admin	Identifier admin	Identity mapping opera- tions	Registry admin	Admin for selected registries
eimAddPolicyAssociation	X	X			X	X
eimAddPolicyFilter	X	X			X	X
eimListPolicyFilters	X	X	X	X	X	X
eimRemove PolicyAssociation	X	X			X	X
eimRemovePolicyFilter						

Table 7. Working with mappings

EIM API	LDAP admin	EIM admin	Identifier admin	Identity mapping opera- tions	Registry admin	Admin for selected registries
eimGetAssociatedIdentifier	X	X	X	X	X	X
eimGetTargetFromIdentifier	X	X	X	X	X	X
eimGetTargetFromSource	X	X	X	X	X	X

Table 8. Working with access

EIM API	LDAP admin	EIM admin	Identifier admin	Identity mapping operations	Registry admin	Admin for selected registries
eimAddAccess	X	X				
eimListAccess	X	X				
eimListUserAccess	X	X				
eimRemoveAccess	X	X				
eimQueryAccess	X	X				

Chapter 3. Migration considerations

This chapter discusses migration issues. Your plan for migrating to a new level of EIM should include information from a variety of sources. These sources of information describe topics such as coexistence service, hardware and software requirements, migration actions, and interface changes. This section should be referenced early in the planning stages as you migrate from one release to another.

Migration from release to release

Migration from EIM Release 6

There are special considerations for applications that will be shared between different releases of z/OS. Prior to z/OS V1R7, EIM application needed to be APF authorized. For z/OS V1R7 or later, the application should not be APF authorized. The one exception is when the application is shared between a down level system and a z/OS V1R7 or later system. The application must remain APF authorized in order for it to work on the down level system. For more information, see “Preparing to run an EIM application” on page 80.

Migration from EIM Release 5 - Starting point

EIM domain controller

Before an EIM domain controller can be migrated to the next release, the LDAP administrator must apply the enhanced schema elements to the LDAP directory. When the EIM domain controller is the z/OS Security Server LDAP Directory Server, you will need to migrate to z/OS Version 1 Release 6 which contains the new schema elements.

For more information on the EIM domain controller, see “Planning considerations for an EIM domain controller” on page 46. For more information on running EIM on z/OS, see “Steps for installing and configuring the EIM domain controller on z/OS” on page 49.

EIM client applications

Existing EIM lookup and administration applications work unchanged with the enhanced domain controller. However, the new policy and certificate support are not available to those applications. New EIM lookup applications and new administration applications work only with the enhanced domain controller.

For more information on EIM client applications, see “Planning for EIM client applications” on page 39.

Removal of SETROPTS EIMREGISTRY/NOEIMREGISTRY

In z/OS V1R4 and R5 the SETROPTS command was used to make an in-storage copy of the local registry name. In z/OS V1R6, the registry name is no longer kept in storage but retrieved once per EIM application that uses the local registry name. The SETROPTS command is no longer needed to make the in-storage copy. For that reason, the EIMREGISTRY/NOEIMREGISTRY keywords are no longer available on the SETROPTS command.

Chapter 4. Planning for EIM

This chapter provides the information you need to understand the task of implementing EIM on your IBM server platform. This chapter also provides the information you need to:

- Understand the task of implementing EIM
- Determine which skills are required to complete your implementation team and create your own implementation plan

This chapter explores the following topics:

Before you begin: The EIM administrator needs to plan carefully for the EIM domain. Before setting up the domain, consider the following:

- What applications will refer to the EIM domain?
- On what systems will the applications run?
- Which system or application registries need to participate in the domain?
- What identifiers do you need to add?
- What associations need to be added between the identifiers and the user IDs in the registries?

Identifying skill requirements

The implementation of EIM requires the interaction of several software products, each with its own required skills. This means that your team can consist of people from several different disciplines, particularly if you work with a large organization.

This section provides the information you need to determine which skills are required to complete your implementation. These skills are presented as job titles for people who specialize in those skills. For example, a task requiring MVS skills is referred to as a task for a ″z/OS system programmer″. Consequently, if some of your team members have multiple skills you might require fewer individuals to complete your team.

Team members

The following describes the responsibilities and roles involved in administering EIM. It also defines potential team members for installing and configuring prerequisite products, and setting up EIM.

An EIM domain can be administered by the LDAP administrator alone, by an EIM administrator, or this responsibility can be divided so the domain can be administered by all of the EIM administrators. Therefore it is advisable to appoint these administrators early and involve them in your planning.

Tip: EIM administrators play an important role in your organization. The decisions they make when creating associations between an identifier and a user ID in a registry can determine who can access your computer systems and what privileges they have when doing so. IBM recommends that you give this authority to those individuals in whom you have a high level of trust.

The following table lists team members (alphabetically) and the tasks and skills needed for setting up EIM:

Table 9. Roles, tasks, and skills for setting up EIM

Role	Tasks	Required Skills
EIM administrator	Responsibilities include: • Coordinating domain operations • Adding, removing, and changing registries, identifiers, and associations between identifiers and user IDs in registries • Granting and removing access to the data kept within an EIM domain	Knowledge of the EIM administration tool you are using
EIM identifier administrator	Responsibilities include: • Creating identifiers • Modifying identifiers • Adding and removing only administrative and source associations (cannot add or remove target associations)	Knowledge of the EIM administration tool you are using
EIM registries administrator	Responsibilities include: • Managing all registries – Adding and removing only target associations (cannot add or remove administrative or source associations) – Updating registries	Knowledge of: • The registries (such as information dealing with user IDs) • The EIM administration tool you are using
EIM registry *X* administrator	Responsibilities include: • Managing individual registries – Adding and removing only target associations (cannot add or remove administrative or source associations) – Updating registry	Knowledge of: • The particular registry (such as information dealing with user IDs) • The EIM administration tool you are using

Table 9. Roles, tasks, and skills for setting up EIM (continued)

Role	Tasks	Required Skills
LDAP administrator	Responsibilities include: • Installing and configuring LDAP (if not already done) • Customizing LDAP configuration for EIM • Creating an EIM domain • Defining users that can bind with the EIM domain controller • Defining the first EIM administrator (optional)	Knowledge of: • LDAP installation, configuration, and customization • EIM administration tool you are using
User registry administrator	Responsibilities include: • Setting up user profiles • Serving as EIM registry administrator (optional)	Knowledge of: • Tools for administering the user registry • EIM administration tool you are using
System programmer	Responsibilities include installing EIM and other software products	Knowledge of: • System programming skills • Installation procedures for the platform
Application programmer	Writes C/C++ applications using EIM APIs	Knowledge of • Platform • C/C++ programming skills • Compiling programs

Planning for EIM client applications

Before you begin: If you are installing an IBM or vendor-written application that exploits EIM, check the product documentation for the hardware and software prerequisites, the specific installation procedures, and configuration procedures. Generally, the applications that use EIM must run on a system where the EIM APIs and the LDAP client are installed.

The EIM APIs are supported on the following hardware and software platforms:

Table 10. EIM APIs software and hardware prerequisites

EIM APIs	LDAP client	Platform
Included in AIX	AIX V5R2	AIX
Enterprise Identity Mapping (EIM) included in z/OS	Integrated Security Services LDAP server or IBM Tivoli Directory Server LDAP server	z/OS
Included in OS/400 V5R2	OS/400 Directory Services	OS/400
Web download	IBM Directory Server	Linux
Windows 2000	Available by Web download	Windows2000

Tip: If you are writing your own application to use EIM, the table above can provide guidance on which platforms the applications can use.

Planning activities for an EIM application include:

1. Identifying the information that is stored in the EIM domain as well as hardware and software prerequisites. For example, the next couple of sections describes what information needs to go into an EIM domain
2. Using the worksheets for recording the information required by the EIM application you are working with. The EIM administrator can take the information from the worksheets and perform the tasks necessary to set up an EIM domain

Planning for an EIM domain

Planning for EIM application begins with the EIM domain. This domain might represent your entire enterprise, a division, a department, or even an application. Plan for the domain to be shared between many applications in order to gain the maximum benefit from having a centralized repository for mapping information.

When setting up your domain you must:

1. Determine whether or not there is an existing domain to use, or if one should be created
2. Name the domain (you can also provide an optional description)

Record your answers in Table 11.

Table 11. Domain worksheet for creating an EIM domain

Parameter name and description	Customized value
description— A string that provides a description of the object you are acting upon.	
domainDN — The distinguished name of the EIM domain. This consists of: • `ibm-eimDomainName=` • *domainName* — The name of the EIM domain you are creating, for example: `My Domain`. (This could be the name of a company, a department, or an application that uses the domain.) • *parentDN* — The distinguished name for the entry immediately above the given entry in the directory information tree hierarchy, for example: o=ibm,c=us **Example:** `ibm-eimDomainName=MyDomain,o=ibm,c=us`	

Planning for EIM registries

The registries that must be defined in an EIM domain are the ones required by the EIM lookup and administration applications that will be using the domain. The registries can represent operating system registries such as RACF or OS/400, a distributed registry such as Kerberos, or a subset of a system registry that is used exclusively by an application.

Consider the following when planning for your registries:

- An EIM domain can contain registries that exist on any platform. A domain controller on z/OS might contain registries for non-z/OS platforms and an EIM domain controller on a non-z/OS platform might contain a z/OS registry, such as RACF.

- The names given to an EIM registry can represent the type of registry, the system the actual registry is on, its network address or its physical location in your enterprise.
- The number of registries that can be defined in a domain is limited by the size of the LDAP directory server where the EIM domain is stored.

Tips:

- Since there might be many registries to consider (source and target), you can use the following worksheet to accommodate some of your planning tasks, such as recording the registries the EIM application uses. (Note that an application might not use all of the available types of associations.) The worksheet can also be used to record a registry alias used by the application. You can fill out one of these worksheets for each application using the EIM domain.
- The installation and configuration information for the application should tell you what types of registries it requires, whether or not registry aliases are used, and the type of associations between the registry user identities and EIM identifiers the application requires.

Table 12. Registry worksheet to help with planning considerations for EIM registries and associations

Registry name	Registry type	Registry alias	Association types required	Registry description

Developing an identity mapping plan

A critical part of the initial Enterprise Identity Mapping (EIM) implementation planning process requires that you determine how you want to use identity mapping in your enterprise. There are two methods that you can use to map identities in EIM:

Identifier associations

Describe the relationships between an EIM identifier and the user identities in user registries that represent that person. An identifier association creates a direct one-to-one mapping between an EIM identifier and a specific user identity. You can use identifier associations to indirectly define a relationship between user identities through the EIM identifier.

If your security policy requires a high degree of detailed accountability, you may need to use identifier associations almost exclusively for your identity mapping implementation. Because you use identity associations to create one-to-one mappings for the user identities that users own, you can always determine exactly who performed an action on an object or on the system.

Policy associations

Describe a relationship between multiple user identities and a single user identity in a user registry. Policy associations use EIM mapping policy support to create many-to-one mappings between user identities without involving an EIM identifier.

Policy associations can be useful when you have one or more large groups of users who need access to systems or applications in your enterprise where you do not want them to have specific user identities for gaining this access. For example, you maintain a Web application that access a specific internal application. You may not want to set up hundreds or thousands of

user identities to authenticate users to this internal application. In this situation, you may want to configure identity mapping such that all the users of this Web application are mapped to a single user identity with the minimum level of authorization required to run the application. You can do this type of identity mapping by using policy associations.

You may decide to use identifier associations to provide the best control of the user identities in your enterprise while gaining the largest degree of streamlined password management. Or, you may decide to use a mixture of policy associations and identifier associations, where appropriate, while you maintain specific control over user identities for administrators. Regardless of what type of identity mapping you decide best meets your business needs and properly fits your security policy, you need to create an identity mapping plan to ensure that you implement identity mapping appropriately.

To create an identity mapping plan, you need to develop an EIM identifier naming plan and plan EIM associations.

Planning considerations for identifiers

Part of your planning activities for an EIM application focuses on the users of the application who need an identifier in the EIM domain. Consequently, to ease administration it is important that you create unique identifiers. When planning your EIM identity mapping needs, you can create unique EIM identifiers for users of EIM-enabled applications and operating systems in your enterprise when you want to create one-to-one mappings between user identities for a user. By using identifier associations to create one-to-one mappings you can maximize the password management benefits that EIM provides.

Tip: For a domain that does not contain a large number of identifiers you might be able to use the actual names. However, for a domain containing a large number of identifiers, you can use an employee number for the identifier and use the real name in the identifier as an alias. Using an employee number allows an administrator to add two employees who happen to have the same name.

The naming plan that you develop depends on your business needs and preferences; the only requirement for EIM identifier names is that they be unique. Some companies may prefer to use each person's full, legal name; other companies may prefer to use a different type of data, such as each person's employee number. If you want to create EIM identifier names based on each person's full name, you may anticipate possible name duplication. How you handle potential duplicate identifier names is a matter of personal preference. You may want to handle each case manually by adding a predetermined character string to each identifier name to ensure uniqueness; for example, you might decide to add each person's department number.

As part of developing an EIM identifier naming plan, you need to decide on your overall identity mapping plan. Doing so can help you to decide when you need to be using identifiers and identifier associations versus using policy associations for mapping identities within your enterprise. To develop your EIM identifier naming plan, you can use the work sheet below to help you gather information about the user identities in your organization and to plan EIM identifiers for the user identities. The work sheet represents the kind of information the EIM administrator needs to know when he creates EIM identifiers or policy associations for the users of an application. The identifier description and additional information fields are free-form and can be used to for descriptive information about the user.

Table 13. Identifier worksheet to help with planning considerations for identifiers

Unique name	Identifier alias	Identifier description	Additional information

An application that is written to use EIM may specify an alias that it uses to find the appropriate EIM identifier for the application, which the application may use in turn to determine a specific user identity to use. You need to check the documentation for your applications to determine whether you need to specify one or more aliases for the identifier. The EIM identifier or user identity description fields are free form and can be used to provide descriptive information about the user.

You do not need to create EIM identifiers for all members of your enterprise at one time. After creating an initial EIM identifier and using it to test your EIM configuration, you can create additional EIM identifiers based on your organization's goals for using EIM. For example, you can add EIM identifiers on a departmental or area basis. Or, you can add EIM identifiers as you deploy additional EIM applications.

After you gather the information that you need to develop an EIM identifier naming plan, you can plan associations for your user identities.

Planning considerations for associations

Associations are entries that you create in an EIM domain to define a relationship between user identities in different user registries. You can create one of two types of associations in EIM: identifier associations to define one-to-one mappings and policy associations to define many-to-one mappings. You can use policy associations instead of, or in conjunction with, identifier associations.

The specific types of associations that you choose to create depends on how a user uses a particular user identity, as well as your overall identity mapping plan.

You can create any of the following types of identifier associations:

Target associations
: Associations for users that normally only access this system as a server from some other client system. This type of association is used when an application performs mapping lookup operations.

Source associations
: You define source associations when the user identity is the first one that a user provides to sign on to the system or network. This type of association is used when an application performs mapping lookup operations.

Administrative associations
: You define administrative associations when you want to be able to track the fact that the user identity belongs to a specific user, but do not want the user identity to be available to mapping lookup operations. You can use this type of association to track all the user identities that a person uses in the enterprise.

A policy association always defines a target association.

It is possible for a single registry definition to have more than one type of association depending on how the user registry that it refers to is used. Although there are no limits to the numbers of, or the combinations of, associations that you can define, keep the number to a minimum to simplify the administration of your EIM domain.

Typically, an application will provide guidance on which registry definitions it expects for source and target registries, but not the association types. Each end user of the application needs to be mapped to the application by at least one association. This association can be a one-to-one mapping between their unique EIM identifier and a user identity in the required target registry or a many-to-one mapping between a source registry of which the user identity is a member and the required target registry. Which type of association you use depends on your identity mapping requirements and the criteria the application provides.

Tips:

- Only add those associations to an EIM domain that are required by the EIM applications that are using the domain. There might be some user IDs in a physical registry that don't have mappings within an EIM domain.

  The number of associations that can be defined in a domain is limited by the size of the LDAP directory server where the EIM domain is stored. There is no hard limit to the number of associations that can be defined between an identifier and user IDs, and between a user ID and identifiers.
- While there are no limits to the combinations of associations that can be defined, it is best to keep the number to a minimum to simplify the administration of your EIM domain.
- The application provides guidance on the registry types it expects, the association types required, and if additional information must be defined for target associations.

As you plan for your EIM applications, use the following worksheets as a guideline for the kind of information the EIM administrator needs to set up the associations. For each end user of the application, there needs to be at least one association between their unique identifier and a user ID in the required registry.

Table 14. Example EIM registry definition information planning work sheet to help with planning considerations for EIM associations

Registry definition name	User Registry type	Registry definition alias	Registry description	Association types

Table 15. Example EIM identifier planning work sheet

Unique identifier name	Identifier or user identity description	Identifier alias

Table 16. Example identifier association planning work sheet

Identifier unique name:		
User registry	**User identity**	**Association types**

Table 17. Example planning work sheet for policy associations

Policy association type	**Source user registry**	**Target user registry**	**User identity**	**Description**

Accessing the EIM domain

To access an EIM domain you must:

1. Be able to bind to the EIM domain controller

   You must first determine the appropriate bind mechanism to connect to the domain controller. EIM APIs support several different mechanism for establishing a connection with the EIM domain controller, each providing a different level of authentication and encryption of the connection. The possible choices are:

   Simple Binds
   A simple bind is an LDAP connection where an LDAP client provides a bind distinguished name and a bind password to the server for authentication. The bind distinguished name and bind password are defined by the administrator in the LDAP directory.

   Server authentication with SSL - server side authentication
   An LDAP server can be configured for SSL or TLS secure connection. The LDAP client and server use digital certificates to encrypt the connection. Only the LDAP server is authenticated. A bind distinguished name and password are used to authenticate the end user.

   Client authentication using SSL
   Provides an additional level of authentication. The LDAP server is configured to require both the LDAP client and server to be authenticated before a connection is established. Digital certificates are used by the LDAP client instead of bind distinguished names and passwords, and the connection is encrypted.

   Kerberos authentication
   An LDAP client can be authenticated to the server using Kerberos, which is a trusted third-party, private key, network authentication system.

   The choice of a bind mechanism is based on the level of security required by the EIM application and the authentication mechanisms supported by the LDAP server hosting the EIM domain. The LDAP server might also require additional configuration to enable the desired authentication mechanism. You must check the documentation (for the LDAP server you are using for your domain controller) for details on how to perform the configuration.

2. Make sure that the bind subject is a member of an EIM authority group. Refer to “EIM access control” on page 30 for more information.

   To access the EIM domain, you must belong to an EIM-defined LDAP access control group or be the LDAP administrator. There are several access control groups that are involved in maintaining an EIM domain. Members of the groups have the ability to update or view different portions of the EIM domain. For information on team members for these groups, see “Team members” on page 37.

Tip: Use the worksheet below as a guide when considering the information needed by the application to access the EIM domain. The EIM application provides guidance on the types of bind mechanisms and the EIM authorities it requires of end users. The values entered here are used by the LDAP administrator to define the bind identity to the LDAP directory server, and the EIM administrator to give the bind identity access to the EIM domain. EIM applications that perform lookups typically require EIM mapping operations authority.

Table 18. Bind worksheet to help in planning for accessing the EIM domain

EIM authorities required	Bind identity	Bind mechanism	Reason needed

Planning considerations for an EIM domain controller

Restriction: EIM requirements on LDAP include the following:

- An LDAP directory server that supports the LDAP (Version 3) protocol. It must also understand the following attributes:
  - ibm-entryUUID attribute
  - ibmattributetypes: aclEntry, aclPropagate, aclSource, entryOwner, ownerPropagate, ownerSource
  - New attribute types and object classes for EIM (schema updates)

Table 19 lists the LDAP servers that can be used as an EIM domain controller.

Table 19. Software and hardware worksheet to help in planning for your EIM domain controller

LDAP servers	Operating system	Hardware
IBM Directory Server v5.1	AIX, Linux, Windows 2000	pSeries® or xSeries™
Integrated Security Services LDAP server	z/OS V1R6	zSeries™
IBM Tivoli Directory Server LDAP server	z/OS V1R8	zSeries
OS/400 Directory Services	OS/400	iSeries®

After reviewing the LDAP directory servers available to you, you can record your choice in the work sheet below.

Table 20. Information needed for LDAP administration

Parameter name and description	Customized value

ldapHost — This consists of:
• The string `ldap://` or `ldaps://`
• The host name or IP address
• The port number (this is optional)
Example:
`ldap://some.ldap.host:389`
`ldaps://some.ldap.host`	

Rules:

1. The LDAP server must be configured for your desired bind mechanisms in order for them to operate successfully. Refer to “Accessing the EIM domain” on page 45 for more information.
2. Before an EIM domain controller can be migrated to the next release, the LDAP administrator must apply the enhanced schema elements to the LDAP directory.
3. New EIM lookup applications and administration applications (introduced in z/OS V1R6 or later) will work only with the enhanced domain controller. Existing EIM administration applications continue to work with the enhanced domain controller. However, the new policy and certificate support are not available. For more information regarding EIM client applications, see “Planning for EIM client applications” on page 39.

Planning EIM administration tools

Some basic steps that must be performed in order to set up an EIM domain are:

1. Creating the domain and define the EIM administrator
2. Creating the registries used by the applications
3. Adding the identifiers
4. Adding the associations

The EIM administrator uses the data recorded in the worksheets provided to perform these tasks. These worksheets are located at “Planning for EIM client applications” on page 39.

Currently, IBM offers several tools an administrator can use to manage the content of an EIM domain, such as the iSeries Navigator or the z/OS eimadmin utility. Software vendors might also offer administration tools in the future. More information can be found in product documentation about the hardware and software prerequisites for the tools, installation, and configuration procedures.

Rule: Generally, administration tools must run on a system where the EIM APIs and the LDAP client are installed

If you plan to use the z/OS eimadmin utility, remember it is part of z/OS Integrated Security Services Enterprise Identity Mapping. (The **eimadmin** command is issued from the z/OS UNIX System Services shell.)

Customizing EIM on your operating system

The platform that supports EIM might provide some unique customizations to allow EIM applications to take advantage of operating system specific features. For example, on z/OS the z/OS Security Server RACF provides RACF profiles that allow a security administrator to define the EIM domain used by an application and the necessary bind credentials.

Task roadmap for implementing EIM

Table 21 shows the tasks and associated procedures for implementing EIM on z/OS. These tasks will constitute a major part of your implementation plan. Your implementation plan should include major tasks, responsibile parties, and a realistic estimate of time and effort required. The major tasks for implementing EIM are provided here as a basis for you to build your own plan.

Table 21. Tasks for implementing EIM on z/OS

Tasks	Associated procedures for z/OS
Install and configure the EIM domain controller	Refer to “Steps for installing and configuring the EIM domain controller on z/OS” on page 49.
Install and configure the EIM administration utility	Refer to “Installing and configuring EIM on z/OS” on page 52.
Use RACF commands to set up and tailor EIM	Refer to Chapter 6, “Using RACF commands to set up and tailor EIM,” on page 67.
Ongoing administration tasks	Refer to “Steps for using the eimadmin utility to manage an EIM domain” on page 53.

Chapter 5. Setting up EIM on z/OS

If you are using EIM on z/OS, many of the topics discussed previously are still applicable. This chapter additionally (and specifically) explores:

It also explores topics about ongoing administration, such as:

Steps for installing and configuring the EIM domain controller on z/OS

Before you begin:

1. You will need LDAP skills to complete this procedure. There are two LDAP servers you can use — the Integrated Security Services (ISS) LDAP server or the IBM Tivoli Directory Server (IBM TDS) LDAP server.
2. You will need to refer to *z/OS Integrated Security Services LDAP Server Administration and Use* or *IBM Tivoli Directory Server Administration and Use for z/OS*.

Rule: Also, for the ISS LDAP server, the following requirements must be met:

- APAR OW55078 (PTF UW92346) must be applied.
- LDAP must be configured to use the TDBM backend.
- The SDBM (RACF) backend is optional.

For the IBM TDS LDAP server, LDAP must be configured to use either the TDBM or the LDBM backend.

1. First, perform the following steps to install and configure LDAP:
   a. Use the following tables to decide what you first need to do. If you are using the ISS LDAP server, refer to Table 22 on page 50. If you are using the IBM TDS LDAP server, refer to Table 23 on page 50.

Table 22. EIM installation and configuration overview for the ISS LDAP server

If ...	Then...	Notes
You do not have LDAP installed and configured...	Follow the instructions in the Administration section of *z/OS Integrated Security Services LDAP Server Administration and Use* to configure the TDBM backend.	The schemas `schema.IBM.ldif` and `schema.user.ldif` need to be loaded.
You have LDAP installed and configured for the SDBM backend but not the TDBM backend...	Follow the instructions in the Administration section of *z/OS Integrated Security Services LDAP Server Administration and Use* to configure the TDBM backend.	The schemas `schema.IBM.ldif` and `schema.user.ldif` need to be loaded.
You have LDAP installed and configured for the TDBM backend...	Go to the next step.	This assumes you have loaded `schema.IBM.ldif` and `schema.user.ldif`.
You plan to use RACF user IDs and passwords to bind within the EIM domain controller...	Follow the instructions in the Administration section of *z/OS Integrated Security Services LDAP Server Administration and Use* to configure the SDBM backend for EIM.	

Table 23. EIM installation and configuration overview for the IBM TDS LDAP server

If ...	Then...	Notes
You do not have LDAP installed and configured...	Follow the instructions in the Administration section of *IBM Tivoli Directory Server Administration and Use for z/OS* to configure the TDBM or LDBM backend.	The schemas `schema.IBM.ldif` and `schema.user.ldif` need to be loaded.
You have LDAP installed and configured for the SDBM backend but not the TDBM or LDBM backend...	Follow the instructions in the Administration section of *IBM Tivoli Directory Server Administration and Use for z/OS* to configure the TDBM or LDBM backend.	The schemas `schema.IBM.ldif` and `schema.user.ldif` need to be loaded.
You have LDAP installed and configured for the TDBM or LDBM backend...	Go to the next step.	This assumes you have loaded `schema.IBM.ldif` and `schema.user.ldif`.
You plan to use RACF user IDs and passwords to bind within the EIM domain controller...	Follow the instructions in the Administration section of *IBM Tivoli Directory Server Administration and Use for z/OS* to configure the SDBM backend for EIM.	

Perform the following steps for the decision you have made.

b. The LDAP server must be configured to accept the different types of bind requests. The information from worksheet Table 18 on page 46 lists the bind mechanisms required by the EIM client applications using this EIM domain controller. See *z/OS Integrated Security Services LDAP Server Administration and Use* or *IBM Tivoli Directory Server Administration and Use for z/OS* for more details.

c. Start the z/OS LDAP server as described in *z/OS Integrated Security Services LDAP Server Administration and Use* or *IBM Tivoli Directory Server Administration and Use for z/OS*.

d. Load the schema definitions.

Rule: If you are migrating from a pre-z/OS Version 1 Release 4 ISS LDAP server, `schema.IBM.ldif` must be loaded. Refer to *z/OS Integrated Security Services LDAP Server Administration and Use* for migration considerations that apply.

Attention:

- An EIM domain must be updated using the EIM APIs or administrative applications that use the EIM APIs. IBM does not recommend using the LDAP utilities and LDAP client APIs to update information in an EIM domain.
- Do not alter the EIM schema definitions unless directed to do so by your IBM Service representative during problem diagnosing.

Restriction: z/OS LDAP by default has a 511–character limit on the length of a distinguished name for an entry. If this default length is exceeded, message ITY0023 (indicating an unexected LDAP error) is issued, indicating that DB2® needs to be reconfigured to support longer distinguished names. This error might show up when working with long identifier, registry, domain names or suffixes. See *z/OS Integrated Security Services LDAP Server Administration and Use* or *IBM Tivoli Directory Server Administration and Use for z/OS* for more details.

2. Second, consider the options you have for setting up an EIM domain that includes z/OS:
   a. Use LDAP on z/OS as the domain controller. (z/OS and non-z/OS applications could access the data.) If using the ISS LDAP server, it must be configured with the TDBM backend. If using the IBM TDS LDAP server, it must be configured with either the TDBM or LDBM backend. If you plan to use RACF user IDs and passwords for the bind credentials, configure the server with the SDBM and the TDBM backends.
   b. Set up the z/OS LDAP server in multi-server mode. This configuration has multiple LDAP servers sharing the same TDBM or LDBM backend store, which is useful if you want to balance the work load between your LDAP servers.
   c. The z/OS EIM application can access a domain controller that resides on another platform.

   Figure 12 on page 52 represents a basic z/OS configuration. Be aware that the WLM support shown in this figure is available only with the ISS LDAP server. It is not available with the IBM TDS LDAP server.

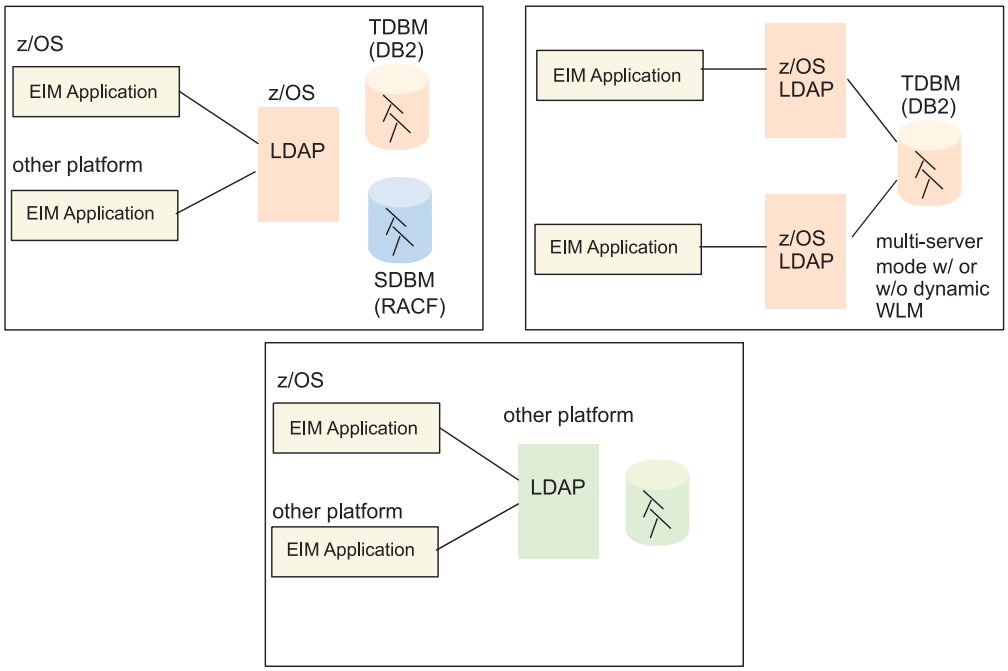


Figure 12. EIM configurations involving z/OS

Installing and configuring EIM on z/OS

Your z/OS system programmer uses SMP/E to install EIM into an HFS directory. By default, EIM is installed in the `/usr/lpp/eim` directory, but your system programmer can determine whether to change the default for these directories.

Table 24 lists important directories for EIM installation. Your system programmer should review the rightmost column of this table, crossing out any defaults that have changed and recording the correct directory names.

Tip: An EIM administrator who uses the eimadmin utility might desire that the directory for the eimadmin utility be placed in the PATH environment variable. This enables the ability to run the utility without having to specify the path when issuing the command (or changing to the `/usr/lpp/eim/bin` directory prior to issuing the command). The PATH enviroment variable can be modified to include the EIM programs directory by issuing the following command from a shell prompt:

```
export PATH=$PATH:/usr/lpp/eim/bin
```

This adds the EIM programs directory to the end of the list of directories to search for programs. Add the export command to a user's `.profile` file so that each time the user enters a shell, the PATH is updated.

Table 24. HFS install directories

Directory and description	Default value or customized value
Main install directory	`/usr/lpp/eim`

Table 24. HFS install directories (continued)

Directory and description	Default value or customized value
EIM library directory; contains the runtime library(eim.dll) and the definition side deck file(eim.x) for linking EIM applications. Also contains the EIM Java Interface classes (eim.jar), zOS Provider classes (eimzOS.jar) Javadoc for EIM Java (eimzOS_DOC.jar) and routines used by EIM Java (libeimJNI.so) **Note:** These files are also symbolically linked in the /usr/lib directory.	`/usr/lpp/eim/lib`
Message catalog directories **Note:** Files in C directory are symbolically linked to the En_US.IBM-1047 directory message catalog files. There are additional symlinks of the En_US.IBM-1047 message catalog files in the /usr/lib/nls/msg/C and /usr/lib/nls/msg/En_US.IBM-1047 directories. Additionally, there are symbolic links to the message catalog files in the Ja_JP directory in the /usr/lib/nls/msg/Ja_JP directory	`/usr/lpp/eim/lib/nls/msg/En_US.IBM-1047` , `/usr/lpp/eim/lib/nls/msg/C`, and `/usr/lpp/eim/lib/nls/msg/Ja_JP`
C/C++ header files for the EIM API prototypes, defined data types, and message catalog constants **Note:** The header files are also symbolically linked in the /usr/include directory.	`/usr/lpp/eim/include`
EIM programs directory (which is where the eimadmin utility program is located)	`/usr/lpp/eim/bin`
EIM man page directory **Note:** There is a symbolic link to the man page in the `/usr/man/C/cat1` and `/usr/man/En_US.IBM-1047/cat1` directories.	`/usr/lpp/eim/man/En_US.IBM-1047/cat1` and `/usr/lpp/eim/man/C/cat1`

Steps for using the eimadmin utility to manage an EIM domain

Before you begin:

This section provides an example of issuing the **eimadmin** command to perform tasks such as:

- Creating an EIM domain
- Granting administration authority
- Adding registries
- Adding enterprise identifiers
- Defining associations

You need to be familiar with this command. Refer to Chapter 9, “The eimadmin utility,” on page 109 and familiarize yourself with the **eimadmin** command.

Note: The eimadmin utility can manage an EIM domain in a z/OS or non-z/OS EIM domain controller. Refer to for “Planning for EIM client applications” on page 39 for more information.

You can perform the following steps to create and manage an EIM domain using the eimadmin utility.

Before you begin:

- The eimadmin utility examples can be entered from the z/OS UNIX System Services shell by an EIM administrator.

- For improved readability each command option is shown on a separate line.
- In most cases you specify multiple options on a single line, separating them with one or more spaces.
- If necessary, you can use the backslash (\) continuation character to break the command into multiple lines.
- The access authority required for successful completion depends on the particular eimadmin operation you specify, and is determined by the bind credential you specify for LDAP authentication. The distinguished name that LDAP associates with the credential should be a member of one or more EIM access groups, which define access authority to EIM data. Refer to "Domain authentication methods" on page 57 for a description of supported bind methods.

To create the domain:

1. Create an EIM domain by entering a command such as the following from the z/OS shell:

   ```
   eimadmin
   -aD
   -d domainDN
   -n description
   -h ldapHost
   -b bindDN
   -w bindPassword
   ```

   The *bindDN* must be the distinguished name for the LDAP administrator. (The description is optional.)

   Example: The following command creates the EIM domain ″My Domain″:

   ```
   eimadmin
   -aD
   -d 'ibm-eimDomainName=My Domain,o=ibm,c=us'
   -n 'An EIM Domain'
   -h ldap://some.ldap.host
   -b 'cn=ldap administrator'
   -w secret
   ```

   Note: This assumes that the ″o=ibm,c=us″ objects are defined in the LDAP Directory. If these objects are not defined, refer to "Example for creating LDAP suffix and user objects" on page 418 for assistance in defining these objects if necessary.

2. Give an administrator EIM administrator authority to the domain by entering a command such as the following from the z/OS shell:

   ```
   eimadmin
   -aC
   -d domainDN
   -c ADMIN
   -q accessUser
   -f accessUserType
   -h ldapHost
   -b bindDN
   -w bindPassword
   ```

   The parameter following `-c` is the *accessType* parameter. In this situation, the value must be `ADMIN`. The *bindDN* must be the distinguished name for the LDAP administrator.

   Tip: If you plan on dividing the administration responsibilities, repeat this command for the other administrative users.

Example: The following command can be issued by the LDAP administrator to give EIM administrator, ″cn=eim administrator,ou=dept20,o=ibm,c=us″, authority to adminster the EIM domain:

```
eimadmin
-aC
-d 'ibm-eimDomainName=My Domain,o=ibm,c=us'
-c ADMIN
-q 'cn=eim administrator,ou=dept20,o=ibm,c=us'
-f DN
-h ldap://some.ldap.host
-b 'cn=ldap administrator'
-w secret
```

Note: This assumes that the ″cn=eim administrator,ou=dept20,o=ibm,c=us″ is defined in the LDAP Directory. If this object is not defined, refer to “Example for creating LDAP suffix and user objects” on page 418 for assistance in defining these objects if necessary.

3. Add registries to the EIM domain by entering a command such as the following from the z/OS shell:

```
eimadmin
-aR
-d domainDN
-r registryName
-y registryType
-n description
-h ldapHost
-b bindDN
-w bindPassword
```

Note: The –y parameter specifies registry type. (The description is optional.) See page 119 for details.

Examples:

The following command adds a RACF registry to the EIM domain named ″My Domain″:

```
eimadmin
-aR
-d 'ibm-eimDomainName=My Domain,o=ibm,c=us'
-r 'RACF Pok1'
-y RACF
-n 'the RACF Registry on Pok System 1'
-h ldap://some.ldap.host
-b 'cn=eim administrator,ou=dept20,o=ibm,c=us'
-w secret
```

The following command adds an OS/400 registry to the EIM domain named ″My Domain″:

```
eimadmin
-aR
-d 'ibm-eimDomainName=My Domain,o=ibm,c=us'
-r 'OS400 RCH1' -y OS400
-n 'the OS400 Registry on Rochester System 1'
-h ldap://some.ldap.host
-b 'cn=eim administrator,ou=dept20,o=ibm,c=us'
-w secret
```

4. Add enterprise identifiers to the domain by entering a command such as the following from the z/OS shell:

```
eimadmin
-aI
-d domainDN
-i identifier
```

```
-n description
-h ldapHost
-b bindDN
-w bindPassword
```

- You can add identifiers at any time after creating the domain.
- The preceding command adds a single identifier to the domain. Alternately, you can add multiple identifiers by specifying a file name as standard input to the eimadmin utility. Specifying a file name indicates using the file of identifiers as input for batch processing of multiple identifiers.

Repeat Step 4 on page 55 as needed.

The *bindDN* must have EIM administrator authority or EIM Identifier administrator authority.

The following command can be issued by the EIM administrator add to an EIM identifier to the domain `My Domain`:

```
eimadmin
-aI
-d 'ibm-eimDomainName=My Domain,o=ibm,c=us'
-i 'John Adam Day'
-h ldap://some.ldap.host
-b 'cn=eim administrator,ou=dept20,o=ibm,c=us'
-w secret
```

5. Create associations between registry user IDs and identifiers by entering commands from the z/OS shell (One or more of the association types, **-t** source, **-t** target, **-t** admin, are required on the command.):

```
eimadmin
-aA
-d domainDN
-r registryName
-u userid
-i identifier
-t admin
-t source
-t target
-h ldapHost
-b bindDN
-w bindPassword
```

The following command creates associations between the user ID JD in the `RACF Pok1` registry:

```
eimadmin
-aA
-d 'ibm-eimDomainName=My Domain,o=ibm,c=us'
-r 'RACF Pok1' -u JD -i 'John Day' -t source -t target
-h ldap://some.ldap.host -b 'cn=eim administrator,ou=dept20,o=ibm,c=us'
-w secret
```

After you enter these commands, you can use the domain for lookup operations. For the preceding examples, the only user mappings available are mappings from `JD` to `JOHNDAY` and from `JOHNDAY` to `JD`.

Notes:

a. You can create associations only after registries and identifiers are in place.

b. The command creates only two associations. Conversely, you can create multiple associations by specifying a file name as standard input to the eimadmin command. Specifying a file name indicates using a file of associations as input for batch processing of multiple associations.

Repeat Step 5 as needed.

6. Give users lookup access to the EIM domain.

```
eimadmin
-aC
-d domainDN
-c MAPPING
-q accessUser
-f DN
-h ldapHost
-b bindDN
-w bindPassword
```

The eimadmin utility allows you to grant access one user at a time or a list of users can be provided in a file using the following command:

```
eimadmin
-aC
-d domainDN
-c MAPPING
-h ldapHost
-b bindDN
-w bindPassword <input-fileName
```

The file must contain a label line following by at least one user name. For example, a bind distinguished name, and the type of the user name.

```
CU                                          ;CS          ;
cn=John Day,c=us                             DN
```

The following command can be issued by the EIM administrator add to give the end user John Day mapping (lookup) authority to the domain `My Domain`:

```
eimadmin
-aC
-d 'ibm-eimDomainName=My Domain,o=ibm,c=us'
-c MAPPING
-q 'cn=John Day,c=us'
-h ldap://some.ldap.host
-b 'cn=eim administrator,ou=dept20,o=ibm,c=us'
-w secret
```

Domain authentication methods

Authentication occurs when an EIM application connects (binds) to the EIM domain controller. z/OS EIM supports the following three authentication methods recognized by LDAP:

- Simple (with or without CRAM-MD5 password protection)
- Digital certificate
- Kerberos

Your LDAP server configuration and security requirements determine which method you choose. The examples in this section illustrate how you can use these methods with the eimadmin utility.

This information explains how the bind credentials specified correspond to the distinguished name that LDAP uses for access checking. Your access to EIM data is determined by the authority groups of which the distinguished name is a member. The exception is the distinguished name for the LDAP administrator that has unrestricted access.

Using simple binds

A distinguished name and password are sufficient credentials for a SIMPLE eimadmin connect type.

```
eimadmin
-lD
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
```

```
-h ldap://some.ldap.host
-S SIMPLE
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Note: Unless an SSL session has been established, the password is sent over the network in plain text, making this method the least secure. The distinguished name that you specify is the one LDAP uses for access checking.

Using CRAM-MD5 password protection

You can use CRAM-MD5 for simple authentication without sending the bind password over the network in plain text, provided both client and server support the method. In the utility command, specify the connect type CRAM-MD5 to indicate simple authentication with password protection.

```
eimadmin
-lD
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-S CRAM-MD5
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Using digital certificates

To bind using a digital certificate, specify the EXTERNAL connect type on the eimadmin command. Ensure the host name identifies a secure host:port value prefixed with *ldaps://*.

Rules:

- You must also specify the name of either a key database file or RACF key ring that contains your client certificate.
- You must specify the label for that certificate if it is not the defined default.
- If you specify a key database file but not its password, the utility prompts you for it.

```
eimadmin
-lD
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldaps://secure.ldap.host
-S EXTERNAL
-K client.kdb
-P clientpw
-N eimadmincert
```

Note: LDAP uses the client certificate's subject distinguished name for access checking.

Using Kerberos

To bind using a Kerberos identity, specify connect type GSSAPI on the eimadmin command. No other credential information is required, but the default Kerberos credential must have been established through a service such as kinit prior to entering the command.

```
kinit eimadministrator@realm.com

eimadmin
-lD
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-S GSSAPI
```

For access checking, LDAP considers a distinguished name formed by prefixing the Kerberos principal name with ″ibm-kgn=″ or distinguished names located through special mapping or searches. See *z/OS Integrated Security Services LDAP Server Administration and Use* for more information.

Using Secure Sockets Layer (SSL)

You can establish an SSL connection along with any of the supported authentication types if your domain controller is configured as a secure host enabled for server authentication.

A secure host is required for EXTERNAL connect.

The strength of SSL is that data transferred over the connection is encrypted, including the password for a SIMPLE bind. The eimadmin utility recognizes the need for an SSL connection when you specify an LDAP host name prefixed `ldaps://`. It then requires that you specify a RACF key ring, or a key database file and its password.

Installation considerations for applications

EIM applications on z/OS may require authority to RACF profiles. See “Preparing to run an EIM application” on page 80 for more information.

Configuration considerations for enabling remote services

To enable EIM to work with remote services such as identity cache and other forms of remote authorization, additional configuration steps are necessary. See “Configuring your environment to use the z/OS Identity Cache” on page 425 and “Configuring the IBM Tivoli Directory Server for remote services support” on page 432 for details on these steps.

Ongoing administration

This section explains how to perform additional administration tasks:

- Managing registries
  - Adding a system and application registry
  - Removing a registry
- Assigning an alias
  - Assigning an alias
  - Removing an alias
  - Assigning an alias to a different registry
- Adding a new user
  - Adding an identifier
  - Adding associations
- Removing a user
  - Removing associations
  - Removing an identifier
- Changing access authority
  - Adding access
  - Removing access

Managing registries

A domain typically contains multiple registries. User identities for a particular system are associated with a system registry, while a subset of identities might be associated with an application registry.

Adding a system and application registry

Create a system registry by entering the following command:

```
eimadmin
-aR
-r 'RACF Pok1'
-y racf
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Enter the following command to define an application registry that is dependent on a previously-defined system registry:

```
eimadmin
-aR
-r 'App1'
-y racf
-g 'RACF Pok1'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Note: Once you define an application registry, you can refer to it by name in EIM APIs and eimadmin commands without having to identify it as an application-type registry.

Listing a registry

You can list any registry using a command similar to the following:

```
eimadmin
-lR
-r 'App1'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Removing a registry

To remove a registry, issue the following command:

```
eimadmin
-pR
-r 'App1'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

All associations linked to the registry are automatically deleted.

Attention: EIM refuses to remove a system registry if any application registries depend on it.

1. You can find the dependents that you must remove by searching for all occurrences of the system registry name in the output from the following command, which lists all registries:

```
eimadmin
-lR
-r '*'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

2. With caution, you can use the '-s rmdeps' option of eimadmin to remove dependent application registries automatically when removing the system registry.

```
eimadmin
-s rmdeps
-pR
-r 'RACF Pok1'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Working with registry aliases

You can define alias names to facilitate registry administration. By establishing aliases that applications use to look up actual registry names, you can make non-disruptive registry changes by managing alias assignments.

Rule: When defining or referencing a registry alias, you must specify an associated registry type. You can use one of the suggested types (refer to "eimChangeRegistryAlias" on page 207) or invent your own.

Assigning an alias

Enter the following command to assign an alias name to an existing registry:

```
eimadmin
-mR
-r 'RACF Test Pok1'
-x 'z/OS' -z 'RACF'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

This example defines the alias 'z/OS' (of type 'RACF') for registry 'RACF Test Pok1'.

Listing an alias

You can list the registry and its aliases using the following command:

```
eimadmin
-lR
-r 'RACF Test Pok1'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Removing an alias

You can delete an alias for a registry using the following command:

```
eimadmin
-eR
-r 'RACF Test Pok1'
-x 'z/OS' -z 'RACF'
```

```
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

This example removes the alias z/OS' (of type 'RACF') for registry 'RACF Test Pok1'.

Assigning an alias name to a different registry

To assign an alias name to a different registry, add the alias name and type to the registry attributes as shown in the example for adding an alias name to a registry above.

Multiple registries can have the same registry alias values. However, if you want the alias to map to a single registry, you must remove that alias from registries in which is was previously defined.

Enter the following two commands to reassign alias 'z/OS' from registry 'RACF Test Pok1' to registry 'RACF Pok1':

```
eimadmin
-mR
-r 'RACF Pok1'
-x 'z/OS' -z 'RACF'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

```
eimadmin
-eR
-r 'RACF Test Pok1'
-x 'z/OS' -z 'RACF'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Adding a new user

You can create an new EIM identifier to represent a new person entering your enterprise. As the person is given access to each system or application through its user registry, you can define an EIM association between the EIM identifier and the corresponding registry defined in EIM.

Adding an identifier

When you create a new EIM identifier, it is assigned a name that is unique within the domain.

The eimadmin utility requires that you specify a unique name (unlike the eimAddIdentifier API option that generates a unique name for you).

You can assign an alternate name, or alias, to multiple identifiers. This non-unique name can be used to further describe the represented individual or to serve as an alternate identifier for lookup operations.

Enter the following command to add a new identifier 'John S. Day' with two aliases:

```
eimadmin
-aI
-i 'John S. Day'
-j '654321'
-j 'Contractor'
```

```
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

You can list the new identifier using the unique name.

The utility returns one entry only.

```
eimadmin
-lI
-i 'John S. Day'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

You can also list the new identifier using an alias name.

The utility returns all entries having 'Contractor' defined as an alternate name.

```
eimadmin
-lI
-j 'Contractor'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Adding associations

You can register the system and application user IDs assigned to the individual by defining EIM associations between the identifier and the corresponding registries.

Enter the following command to create source and target associations for user ID 'JD' in registry 'RACF Pok1':

```
eimadmin
-aA
-i 'John S. Day'
-r 'RACF Pok1'
-u 'JD'
-t source
-t target
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Listing associations

Enter the following command to list all associations for 'John S. Day':

```
eimadmin
-lA
-i 'John S. Day'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Removing a user

To completely erase a person's identity from your EIM domain, remove the identifier.

If you only need to reflect the deletion of a user ID from a registry, simply remove the corresponding EIM associations.

Removing associations

Enter the following command to remove the source and target associations for user ID 'JD' in registry 'RACF Pok1':

```
eimadmin
-pA
-i 'John S. Day'
-r 'RACF Pok1'
-u 'JD'
-t source
-t target
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Removing an identifier

Enter the following command to remove an identifier and its associations, including identifier aliases:

```
eimadmin
-pI
-i 'John S. Day'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Changing access authority

A user is permitted to perform EIM administrative or lookup operations based on the authority groups containing the user's LDAP distinguished name (DN). The user's DN is determined by the credentials authenticated when connecting to LDAP.

Suppose a user has registry administrator authority over a specific registry and your task is to switch the user's authority to a different registry. You can accomplish this task in two steps:

1. Adding the user to the new registry adminstrator group
2. Removing the user from the prior group

Adding access authorities

Enter the following command to add user DN 'cn=Reggie King,ou=dept20,o=ibm,c=us' to the registry administration group for 'RACF Pok1':

```
eimadmin
-aC
-q 'cn=Reggie King,ou=dept20,o=ibm,c=us'
-f DN
-c registry
-r 'RACF Pok1'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Listing access authorities

Enter the following command to list all EIM access authorities for the user:

```
eimadmin
-lC
-q 'cn=Reggie King,ou=dept20,o=ibm,c=us'
-f DN
```

```
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Removing access authorities

Enter the following command to remove the user from the prior registry administration group for 'RACF Test Pok1':

```
eimadmin
-pC
-q 'cn=Reggie King,ou=dept20,o=ibm,c=us'
-f DN
-c registry
-r 'RACF Test Pok1'
-d 'ibm-eimDomainName=MyDomain,o=ibm,c=us'
-h ldap://some.ldap.host
-b 'cn=eimadministrator,ou=dept20,o=ibm,c=us'
-w secret
```

Chapter 6. Using RACF commands to set up and tailor EIM

An EIM administration application creating, deleting, or changing descriptive information about a domain must provide domain name and bind information. This chapter explores:

Using RACF for EIM domain access

The RACF administrator can use RACF commands to do the following:

- Add an EIM domain name and bind information for system-wide use
- Add an EIM domain name and bind information for use by a server
- Add an EIM domain name and bind information for use by an administrative user
- Assign a name to the local RACF registry for use by a lookup application

Tip: Issuing these commands is optional. However, setting up your system this way can eliminate the need for individual applications to handle EIM domain and bind information.

The default domain and bind information can be specified in one of three places:

1. The user ID the application runs under has the name of an LDAPBIND class profile in its USER profile
2. The IRR.EIM.DEFAULTS profile in the LDAPBIND class
3. The IRR.PROXY.DEFAULTS profile in the FACILITY class

These RACF profiles can be set up in such a way as to control the access the application has to the EIM domain:

- New connections with an EIM domain can be enabled or disabled by using keywords on the RDEFINE or RALTER commands.
- Bind credentials can be specific to the server or administrator who uses them.

The EIM APIs try to retrieve the information from a profile if the application does not explicitly supply the information to the EIM APIs using parameters. Applications or other services that use EIM can instruct their callers to define a profile in the LDAPBIND class profile.

Setting up default domain LDAP URL and binding information

Servers that use an EIM domain require the name and location of the EIM domain and the appropriate credentials to bind to the LDAP directory service containing the EIM domain. You can store the EIM domain name, its URL, bind distinguished name, and bind password in RACF profiles. (See Table 25 for the ways a security administrator can set up profiles.)

This section explores:

- "Storing LDAP binding information in a profile"
  - "Adding EIM domain and bind information for servers or administrative users" on page 69
  - "Adding a system default using the IRR.EIM.DEFAULTS profile" on page 69
  - "Adding a system default using the IRR.PROXY.DEFAULTS profile" on page 70

Storing LDAP binding information in a profile

Before you begin:

- Use the following decision table to determine which profile to use:

Table 25. Decision table for RACF profiles

If ...	Then ...
The EIM domain is in the default system LDAP directory ...	Set up the IRR.PROXY.DEFAULTS profile in the FACILITY class. (This is the simplest way to set up a profile.)
A server needs to reference an EIM domain that is not in the system default LDAP directory ... (This could be because the IRR.PROXY.DEFAULTS profile has different bind information than the application using the EIM domain requires.)	Set up a profile in the LDAPBIND class. Add the name of the LDAPBIND class profile to the user profile used by the application.

- **Tip:** You need to know certain information to use as parameters in RACF commands. Refer to the *z/OS Security Server RACF Command Language Reference* for more information. Fill in the missing (Value column) information in the following table:

Table 26. LDAP information needed for creating RACF profiles

Information needed	Where to get it	Value
bindDN —The distinguished name to use for LDAP binding. **Example:** `cn=EIM user,o=ibm,c=us`	From the LDAP administrator	
bindPasswd — The password for LDAP binding. **Example:** `secret`	From the LDAP administrator	Note that this is not something that should be written down.
domainDN — The distinguished name of the EIM domain.	From the EIM administrator	

Table 26. LDAP information needed for creating RACF profiles (continued)

Information needed	Where to get it	Value
ldapHost — The LDAP host. This consists of the: • String `LDAP://` or `LDAPS://` , which specifies the LDAP protocol to use when binding • Host name or IP address • A colon (:) followed by the LDAP port number, such as ″:389″ (This portion of the host name is optional if the LDAP server is using the default port.) **Example:** `LDAP://SOME.LDAP.HOST:389`	From the LDAP administrator	
racfProfileName— The name of the RACF profile that stores the following information when the caller does not provide it: • *ldapHost* • *bindDN* • *bindPasswd* • *domainDN* **Example:** `JOESDOMAIN` **Note:** This profile can be a profile defined in the LDAP bind class, the IRR.EIM.DEFAULTS profile in the LDAPBIND class, or the IRR.PROXY.DEFAULTS profile in the FACILITY class.	Defined by the RACF administrator	

Adding EIM domain and bind information for servers or administrative users

To create a profile for LDAP binding information:

1. Perform the following steps if you are creating a profile in the LDAPBIND class:
   a. To define the domain in the LDAPBIND class, enter:

      ```
      RDEFINE LDAPBIND racfProfileName
      EIM(DOMAINDN(domainDN))
      PROXY(LDAPHOST(ldapHost)
      BINDDN(bindDN) BINDPW(bindPasswd))
      ```

      Notes:

      1) `OPTIONS(ENABLE)` is the default value.

   b. To update the user profile:

      ```
      ADDUSER ASERVER EIM(LDAPPROF(racfProfileName))
      ```

Adding a system default using the IRR.EIM.DEFAULTS profile

1. If you are using the IRR.EIM.DEFAULTS profile in the LDAPBIND class, enter:

   ```
   RDEFINE LDAPBIND IRR.EIM.DEFAULTS
   PROXY(LDAPHOST(ldapHost)
   BINDDN(bindDN) BINDPW(bindPasswd))
   EIM(DOMAINDN(domainDN))
   ```

   Note: `OPTIONS(ENABLE)` is the default value.

Adding a system default using the IRR.PROXY.DEFAULTS profile

If no LDAPBIND class profile is associated with the caller's user profile, the EIM services look for the EIM domain's LDAP URL and binding information in the IRR.EIM.DEFAULTS profile in the LDAPBIND class followed by the IRR.PROXY.DEFAULTS profile in the FACILITY class. For example, the following command sets up the binding information in the IRR.PROXY.DEFAULTS profile in the FACILITY class:

```
RDEFINE FACILITY IRR.PROXY.DEFAULTS
PROXY(LDAPHOST(LDAP://SOME.BIG.HOST:389)
BINDDN('cn=Joes Admin,o=ibm,c=us') BINDPW(secret))
EIM(DOMAINDN('ibm-eimDomainName=Joes Domain,o=ibm,c=us'))
```

In this case, the domain's LDAP URL is:

```
LDAP://SOME.BIG.HOST:389/ibm-eimDomainName=Joes Domain,o=ibm,c=us
```

Optionally setting up a registry name for your local RACF registry

Many of the EIM APIs require the name of a registry. For example, if you are adding a registry to an EIM domain, you should know the name of the new registry. However, you can use the lookup APIs (such as eimGetTargetFromSource, eimGetIdentifierFromSource, and eimGetAssociatedIdentifiers) to convert:

1. A user ID to its equivalent RACF user ID
2. A local RACF user ID to an enterprise identifier

For such applications, you can eliminate the requirement for providing the RACF registry name or its alias on the local system. You do this by giving a name to the local RACF registry.

Steps for setting up lookups that do not need a registry name

Before you begin: You need to know the registry name:

Table 27. Local registry name needed for creating RACF profiles

Information needed	Where to get it	Value
registryName — The name of the RACF registry. **Example:** `Registry on POK System`	EIM administrator	

Perform the following to set up EIM so that you do not need a registry name on every lookup. To define the local registry, enter the following RACF command in which *registryName* is the name of the local registry:

```
RDEFINE FACILITY IRR.PROXY.DEFAULTS EIM(LOCALREGISTRY(registryName))
```

Note: EIM does not look for the registry name in an LDAPBIND class profile.

You can also configure the system with a kerberos registry name and an X.509 registry name. Issue the following commands to define default kerberos and X.509 registries for the configured EIM domain:

```
RALTER FACILITY IRR.PROXY.DEFAULTS EIM(KERBREGISTRY(registry name)
X509REGISTRY(registry name))
```

This access can be removed with the following command:

```
RALTER FACILITY IRR.PROXY.DEFAULTS EIM(NOKERBREGISTRY NOX509REGISTRY)
```

Note that these registry names need to be defined in the configured EIM domain.

For more information on defining a registry name, refer to “EIM registry definition” on page 13.

Ongoing RACF administration

You might need to perform the following tasks as part of ongoing RACF administration:

- “Disabling use of an EIM domain”
- “Using output from the RACF database unload utility and eimadmin to prime your EIM domain with information”

Disabling use of an EIM domain

You might need to temporarily disable use of a RACF profile with a configured EIM domain or a system-wide default EIM domain. You might want to do this if the EIM information in a domain has been compromised or a security administrator wants to stop the system or server from establishing new connections with the EIM domain. You can use RACF commands to disable a domain without deleting EIM information from the RACF profiles. When an EIM domain is disabled through a RACF profile, existing connections to the domain complete their work. However, if an EIM service is trying to establish a connection with such a domain, the EIM service does not continue to look for an enabled domain.

Steps for disabling use of an EIM domain

Perform the following steps to disable a server from using the configured EIM domain (This applies only to a server that has an ldapbind class profile specified for its user ID):

1. If you want to disable a server (rather than a system) from using a configured EIM domain, enter the following command:

   ```
   RALTER LDAPBIND ldapbind_profile  EIM(OPTIONS(DISABLE))
   ```

Note: No change is required to the user profile.

Tip: To disable a system-wide default EIM domain (rather than a server) that default profiles use, enter one of the following commands:

```
RALTER FACILITY IRR.PROXY.DEFAULTS EIM(OPTIONS(DISABLE))
RALTER LDAPBIND IRR.EIM.DEFAULTS EIM(OPTIONS(DISABLE))
```

Using output from the RACF database unload utility and eimadmin to prime your EIM domain with information

You can start to put EIM information (identifiers, RACF user IDs, and associations) into your EIM domain by using output from DBUNLOAD and eimadmin.

For large installations, priming the EIM domain with identifiers and associations can involve a lot of work. To make the task of getting started with EIM easier, the eimadmin utility accepts as input a file containing a list of identifiers and associations.

The section explores the steps for setting up an EIM domain based on user information contained in a RACF database. The initial assumptions are that the EIM

domain, World Wide Domain, has been created and a SAF system registry, SAF user IDs, is defined in the domain. The ldap host name for the domain is `ldap://some.big.host`. The EIM administrator uses the bind distinguished name of `cn=EIM Admin,o=My Company,c=US` and the password is secret. The EIM administrator bind distinguished name has been given EIM administrator authority and can perform all of the steps below. A user with other types of EIM authority can perform a subset of the steps below:

- EIM identifier administrator authority only works with identifiers and source and target associations
- EIM registries administrator authority only works with target associations
- EIM registry-specific administrator authority for the SAF registry only works with target associations in the SAF registry

1. Request from your RACF security administrator a file containing a copy of the user profiles in the RACF database. The RACF security administrator can:
   a. Run the database unload utility (IRRDBU00) to create the sequential file
   b. Run the file through a sort program, such as DFSORT or DFSORT ICETOOL to extract just the user profiles and desired fields. The User Basic Data Record (0200) contains the user ID and the programmer name. In this example, the programmer name is used for the EIM identifier.

   The DFSORT ICETOOL Report format has a 1-4 character name (for example, EIM). It contains the ICETOOL statements that control report format and record summary information, such as SORT, COPY, DISPLAY, and OCCURS statements. An example of a report format which can be used to extract RACF user IDs and the programmer names associated with the user IDs is below:

   Example:

```
************************************************************************
* Name: EIM                                                            *
*                                                                      *
* Find all user IDs in the RACF database and their name                *
************************************************************************
 COPY    FROM(DBUDATA) TO(TEMP0001) USING(RACF)
 OCCURS  FROM(TEMP0001) LIST(PRINT) -
         TITLE('user IDs and Names') -
         ON(10,8,CH) HEADER('USER ID') -
         ON(79,20,CH) HEADER('Name')
```

   The record selection criteria is as follows:

   - The name of the member containing the record selection criteria is the report member name followed by CNTL (such as EIMCNTL).
   - Record selection is performed using DFSORT control statements, such as SORT and INCLUDE.
   - The SORT command is used to select and sort records.
   - The INCLUDE command is used to specify conditions required for records to appear in the report.

   The following is an example of the record selection criteria that could be used. In this report, we are including only User Base records (record type 0200) which have the RACF user ID and the programmer name. The selection criteria allows records for the special RACF user IDs, irrcerta, irrmulti, and irrsitec, to remain undisclosed in the final report.

```
SORT    FIELDS=(10,8,CH,A)
INCLUDE COND=(5,4,CH,EQ,C'0200',AND,
              (10,8,CH,NE,C'irrcerta',AND,
```

```
             10,8,CH,NE,C'irrmulti',AND,
             10,8,CH,NE,C'irrsitec'))
             10,7,CH,NE,C'IBMUSER',AND,
OPTION  VLSHRT
```

To use the RACFICE procedure to generate the report:

```
//jobname JOB Job card...
//        SET DBUDATA=USER01.TEST.IRRDBU00
//stepname EXEC RACFICE,REPORT=EIM
```

Result: The output from the utility looks like this:

```
User ID               Name
-------------------   --------------------
ANN                   ANN J. AUSTIN
CHRIS                 CHRISTINE IRVING
DENICE                DENICE GARDNER
DIANA                 DIANA MACMILLIAN
ERIC                  ERIC D. ADAMS
JAY                   JASON SWIFT
MAURA                 MAURA FISHER
OMAR                  OMAR ZACHARY
PEGGY                 PEGGY B. WOLF
RANDY                 RANDY BRAUTIGAN
RICH                  RICH CLANCY
ROSS                  ROSS SIMPSON
SCOTT                 SCOTT SMYTHE
SHOZAB                SHOZAB SYED
SHRUTI                SHRUTI MODI
TRACY                 TRACY ROWLINGS
TERRY                 TERRY HAMMER
VIVIAN                VIVIAN BRONTE
WILLY                 WILLIAM TROTSKY
```

Tips:

- *z/OS Security Server RACF Security Administrator's Guide* contains instructions on how to run the database unload utility and use the sort programs.

2. When you receive the report from the security administrator, you should move it to a file in the hierarchical file system (HFS).
3. Add a eimadmin utility ″label line″ to the file containing user profiles. You can use any one of the editors available from the OMVS shell (such as OEDIT).

   The following is an example of the updated file, *racfUsers.txt* with a label line added and the DFSORT column headers commented out.

   Tips Any user IDs that should not go into the EIM domain, such as user IDs belonging to servers, can also be commented out.

```
#User ID              Name
#-------------------  --------------------
UN                 ;  IU                 ;
ANN                   ANN J. AUSTIN
CHRIS                 CHRISTINE IRVING
DENICE                DENICE GARDNER
DIANA                 DIANA MACMILLIAN
ERIC                  ERIC D. ADAMS
JAY                   JASON SWIFT
MAURA                 MAURA FISHER
OMAR                  OMAR ZACHARY
PEGGY                 PEGGY B. WOLF
RANDY                 RANDY BRAUTIGAN
RICH                  RICH CLANCY
ROSS                  ROSS SIMPSON
SCOTT                 SCOTT SMYTHE
SHOZAB                SHOZAB SYED
SHRUTI                SHRUTI MODI
```

```
TERRY                   TERRY HAMMER
TRACY                   TRACY ROWLINGS
VIVIAN                  VIVIAN BRONTE
WILLY                   WILLIAM TROTSKY
```

4. Add identifiers and list the results using the **eimadmin** shell command:

```
eimadmin
-aI
-d "ibm-eimDomainName=World Wide Domain,o=My Company,c=US"
-h ldap://some.big.host
-b "cn=EIM Admin,o=My Company,c=US"
-w secret <racfUsers.txt
```

5. To list the identifiers added above, issue:

```
eimadmin
-lI
-d "ibm-eimDomainName=World Wide Domain,o=My Company,c=US"
-h ldap://some.big.host
-b "cn=EIM Admin,o=My Company,c=US"
-w secret <racfUsers.txt
```

6. Create source and target associations between the identifiers and the user IDs in RACF. Because the file *racfUsers.txt* contains a label line that identifies user IDs as well as unique identifier names, it can be used to create associations:

```
eimadmin
-aA
-t source
-t target
-r"SAF user IDs"
-d "ibm-eimDomainName=World Wide Domain,o=My Company,c=US"
-h ldap://some.big.host
-b "cn=EIM Admin,o=My Company,c=US"
-w secret <racfUsers.txt
```

7. To list the associations added above:

```
eimadmin
-lA
-d "ibm-eimDomainName=World Wide Domain,o=My Company,c=US"
-h ldap://some.big.host
-b "cn=EIM Admin,o=My Company,c=US"
-w secret <racfUsers.txt
```

8. The following **eimadmin** commands can be used to give EIM Mapping Operations authority to each of the users (identified in the file `racfUsersDNs.txt`):

```
eimadmin
-aC
-c MAPPING
-d "ibm-eimDomainName=World Wide Domain,o=My Company,c=US"
-f DN
-h ldap://some.big.host
-b "cn=EIM Admin,o=My Company,c=US"
-w secret <racfUsersDNs.txt
```

   To list the accesses that have been granted, issue:

```
eimadmin
-lC
-c MAPPING
-d "ibm-eimDomainName=World Wide Domain,o=My Company,c=US"
-h ldap://some.big.host
-b "cn=EIM Admin,o=My Company,c=US"
-w secret <racfUsersDNs.txt
```

   Tip: At a minimum, a user who is looking for a mapping in the EIM domain needs to have EIM mapping operations authority. In most cases, the application has one set of credentials for connect with an EIM domain, and those

credentials are shared by all users. However, if individual access is needed, then a bind distinguished name needs to be defined for each of the users and given EIM mapping operations authority.

Suppose the file racfUsersDNs.txt contains this list of bind distinguished names that were defined to the LDAP server containing the EIM domain controller and an eimadmin label line:

```
CU                                      ;
cn=Ann J. Austin,o=My Company,c=US
cn=Chrisine Irving,o=My Company,c=US
cn=Denice Gardener,o=My Company,c=US
cn=Diana MacMillian,o=My Company,c=US
cn=Eric D. Adams,o=My Company,c=US
cn=Jason Swift,o=My Company,c=US
cn=Maura Fisher,o=My Company,c=US
cn=Omar Zachary,o=My Company,c=US
cn=Peggy B. Wolf,o=My Company,c=US
cn=Randy Brautigan,o=My Company,c=US
cn=Rich Clancy,o=My Company,c=US
cn=Ross Simpson,o=My Company,c=US
cn=Scott Smythe,o=My Company,c=US
cn=Shozab Syed,o=My Company,c=US
cn=Shruti Modi,o=My Company,c=US
cn=Terry Hammer,o=My Company,c=US
cn=Tracy Rowlings,o=My Company,c=US
cn=Vivian Bronte,o=My Company,c=US
cn=William Trotsky,o=My Company,c=US
```

Chapter 7. Developing applications

The z/OS UNIX programmer codes customer EIM lookups and administrative applications, integrating calls to the EIM APIs within these applications. The EIM APIs are implemented as ″C″ programming interfaces.

This chapter explores:

Writing EIM applications

Default registry names

Many of the EIM APIs require the specification of the name of a registry. For example, if you are adding a registry to an EIM domain, you should know the name of the new registry being used. However, you might use the lookup APIs (such as eimGetTargetFromSource, eimGetTargetFromIdentifier, and eimGetAssociatedIdentifiers) to convert:

- A user ID to its equivalent RACF user ID
- A local RACF user ID to an enterprise identifier

Tip: For such applications, you can eliminate the requirement for providing the RACF registry name or its alias on the local system. This is done by storing a name for the local RACF registry in the IRR.PROXY.DEFAULTS profile in the FACILITY class.

Defining private user registry types in EIM

Define a private user registry type in EIM

To define a user registry type that EIM is not predefined to recognize, you must specify the registry type in the form of ObjectIdentifier-normalization, where ObjectIdentifier is a dotted decimal object identifier (OID), such as `1.2.3.4.5.6.7`, and normalization is either the value caseExact or the value caseIgnore.

If you need a private OID for use only within your enterprise, you can pick any arbitrary number not already in use. However, private OIDs that you want to use outside your enterprise must be obtained from legitimate OID registration authorities. Doing so ensures that you create and use unique OIDs which helps you avoid potential OID conflicts with OIDs created by other organizations.

There are two ways of obtaining OIDs:

1. Registering your OIDs with an authority is a good choice, for example, when you need a small number of fixed OIDs to represent information. For example, these OIDs might represent certificate policies for users in your enterprise.
2. Obtaining an arc assignment, which is a dotted decimal object identifier range assignment, is a good choice if you need a large number of OIDs, or if your OID assignments are subject to change. The arc assignment consists of the beginning dotted decimal numbers from which you must base your ObjectIdentifier. For example, the arc assignment could be `1.2.3.4.5.`. You could then create OIDs by adding to this basic arc. For example, you could create OIDs in the form `1.2.3.4.5.x.x.x`).

   Tip: You can learn more about registering your OIDs with a registration authority by reviewing these Internet resources:

   - ANSI is the registration authority for the United States for organization names under the global registration process established by ISO and ITU. A fact sheet with links to an application form is located at the ANSI Web site:

     `http://web.ansi.org/public/services/reg_org.html`

     Note: The ANSI OID arc for organizations is `2.16.840.1`. ANSI charges a fee for OID arc assignments. It takes approximately two weeks to receive the assigned OID arc from ANSI. ANSI will assign a number (NEWNUM), creating a new OID arc: `2.16.840.1.NEWNUM`.
   - In most countries or regions, the national standards association maintains an OID registry. As with the ANSI arc, these are generally arcs assigned under the OID 2.16. It might take some investigation to find the OID authority for a particular country or region. The addresses for ISO national member bodies can be found at:

     `http://www.iso.ch/addresse/membodies.html`

     The information includes a postal address and electronic mail, and in many cases a Web site is specified as well.
   - Another possible starting point is the International Register of ISO DCC Network Service Access Point (NSAP) schemes. The registry for schemes can be obtained at:

     `http://www.fei.org.uk/fei/dcc-nsap.htm`

     This Web site currently lists contact information for thirteen naming authorities, some of which also assign OIDs.
   - The Internet Assigned Numbers Authority (IANA) assigns private enterprise numbers, which are OIDs, in the arc `1.3.6.1.4.1`. IANA has assigned arcs to over 7,500 companies to date. The application page is located at:

     `http://www.iana.org/forms.html`

     It can be found under ″Private Enterprise Numbers″. The IANA OID is free and is usually received in about one week. IANA assigns a number (NEWNUM), so the new OID arc will be `1.3.6.1.4.1.NEWNUM`.
   - The U.S. Federal Government maintains the Computer Security Objects Registry (CSOR). The CSOR is the naming authority for the arc 2.16.840.1.101.3, and is currently registering objects for security labels, cryptographic algorithms, and certificate policies. The certificate policy OIDs are defined in the arc `2.16.840.1.101.3.2.1`. The CSOR provides policy OIDs to agencies of the United States Federal Government. For more information about the CSOR, refer to:

     `http://csrc.nist.gov/csor/`

For more information on OIDs for certificate policies, see http://csrc.nist.gov/csor/pkireg.htm.

Building an EIM application

Any user of an EIM application (including eimadmin) needs a z/OS UNIX System Services UID and GID assigned to them. See *z/OS UNIX System Services Planning* for details on how to do this.

This section explores:

- "C/C++ Compile considerations"
- "C/C++ Link-edit considerations" on page 80

Note: The EIM programming interface is provided in a set of C/C++ functions in the EIM DLL and a pair of Java jar files. The DLL is loaded at program run time so that calls to the functions in the interface can be made. In order to compile and link-edit a C/C++ program that uses the EIM API, use the following guidelines.

C/C++ Compile considerations

Put the following include statement in all C or C++ source files that make calls to the EIM programming interface or use EIM data structures.

#include <eim.h>

Note: If defaults were used during EIM installation, the eim.h file is located in the `/usr/lpp/eim/include` directory. The eim.h file has been symbolically linked in the `/usr/include` directory. If EIM was not installed in the default location, you might need to specify the directory where the compiler is to find the eim.h file with the **-I** parameter.

Tip: Specify **-D_EIM_EXT** on the compile of the source files that include eim.h. This ensures full support for the errno values defined for EIM and that they are proper by defined for your application's use. Additionally, the Language Environment library level must be at z/OS release 4 or above. To set the Language Envirnoment library level to the z/OS Version 1 Release 4 level, specify **target(0x41040000)**. (To add this to the C/C++ command, specify **-Wc,target\(0x41040000\)**).

When compiling a program that makes EIM API calls, be sure to specify the DLL option to the compiler. (**-Wc,dll** when using the c89/cc/c++ commands)

Tip: Ensure your application has POSIX(ON) specified so it can use the EIM APIs.

The values returned by the EIM APIs are standard POSIX errnos with five additions. These errnos, including the additions, can be used as input to the `strerror()` or `perror()` functions:

- EBADDATA = Data is not valid
- EUNKNOWN = Unknown system state
- ENOTSUP = Operation not supported
- EBADNAME = The object name specified is not correct
- ENOTSAFE = The function is not allowed

Refer to "eimErr2String" on page 241 for more information.

C/C++ Link-edit considerations

Tip: When link-editing, be sure to specify the EIM ″exports″ file in the set of files to be link-edited with the program.

The EIM export file (eim.x) is located in the library directory of the EIM install directory, which is `/usr/lpp/eim/lib` by default. For convenience, a symbolic link to this file has been created in the `/usr/lib` directory. If the default directory was used during EIM installation, the export file could be specified as `/usr/lib/eim.x` or `/usr/lpp/eim/lib/eim.x`.

Preparing to run an EIM application

Tip: When running an EIM application, be sure that the EIM DLL is accessible by ensuring the LIBPATH environment variable includes `/usr/lib`. Be sure that the directory your programs are located in are in the PATH environment variable.

Since the EIM message catalogs are symbolically linked in the `/usr/lib/nls/msg` directories, it should not be necessary to update NLSPATH.

Note: previous releases of z/OS required programs that used the EIM APIs to be APF-authorized. This requirement no longer exists. This means that you must remove the APF-authorization extended attribute for each existing EIM application program (new applications you add for this release are not affected). This attribute is removed by using the **extattr –a** command.

For example **extattr –a eimprog** would remove the APF-authorization bit for the program **eimprog** in the current directory. For more information on the **extattr** command, refer to *z/OS UNIX System Services Command Reference* or *z/OS UNIX System Services Planning*.

Accessing RACF profile checks

If your C/C++ or Java application is retrieving configuration information from RACF profiles, for example:

- the default EIM domain dn, ldap host, bind dn, and bind password
- the default registry names

then the RACF user ID of the application callers may need authority to SAF services that allow the information to be retrieved from the RACF database.

The following table lists the access requirements for the EIM APIs when they need to obtain information configuration information from a RACF profile.

Table 28. EIM API access requirements

EIM C/C++ API	SAF Callable service used by the EIM APIs	Authorization requirements for the calling application.
eimCreateHandle, eimGetAssociatedIdentifiers eimGetTargetFromIdentifier eimGetTargetFromSource eimRetrieveConfiguration	R_GetInfo	The calling application can be running in system key or supervisor state or one of the following: • The RACF user ID of the caller's address space has READ authority to the BPX.SERVER profile in the FACILITY class • The current RACF user ID has READ authority to the IRR.RGETINFO.EIM profile in the FACILITY class The FACILITY class must be active and RACLISTed before unauthorized (problem program state and keys) will be granted the authority to use this SAF service.
eimConnect, eimConnectToMaster	R_dcekey	The calling application can be running in system key or supervisor state or one of the following: • The RACF user ID of the caller's address space has READ access to the BPX.SERVER profile in the FACILITY class • The current RACF user ID has READ authority to the IRR.RDCEKEY profile in the FACILITY class Applications that are not authorized (problem program state and keys) must be program controlled (extattr +p), the address space must be clean and the FACILITY class must be active and RACLISTed before the application will be granted authority to use this SAF service.

Table 28. EIM API access requirements (continued)

EIM C/C++ API	SAF Callable service used by the EIM APIs	Authorization requirements for the calling application.
eimSetConfigurationExt	R_GetInfo R_admin	The calling application can be running in system key or supervisor state or one of the following: • The RACF user ID of the caller's address space has READ authority to the BPX.SERVER profile in the FACILITY class • The current RACF user ID has READ authority to the IRR.RGETINFO.EIM profile in the FACILITY class For applications that are not authorized (problem program state and keys), the current RACF user ID must satisfy the following requirements: • Have READ authority to the following profiles in the FACILITY class: – IRR.RADMIN.ALTUSER – IRR.RADMIN.RDEFINE – IRR.RADMIN.RALTER – IRR.RADMIN.RDELETE • Have authority to issue the following commands: – ALTUSER – RALTER – RDEFINE – RDELETE The FACILITY class must be active and RACLISTed before the application will be granted authority to use this SAF service

Special considerations for applications that will be shared between different releases of z/OS

Prior to z/OS V1R7, the EIM application needed to be APF authorized. For z/OS V1R7 or later, the application should not be APF authorized. The one exception is when the application is shared between a down level system and a z/OS V1R7 or later system. The application must remain APF authorized in order for it to work on the down level system. For more details on APF authorization, see "Preparing to run an EIM application" on page 80.

APIs for retrieving the LDAP URL and binding information

EIM APIs, eimCreateHandle, eimConnect, and eimConnectToMaster, use SAF APIs to retrieve the domain's LDAP URL and binding information from RACF profiles when the caller does not provide them. The order of search for the domain and bind information is:

1. As input parameters on the call to the EIM API
2. In profiles as follows:
   a. LDAPBIND profile named in the EIM segment of the caller's USER profile
   b. IRR.EIM.DEFAULTS profile in the LDAPBIND class
   c. IRR.PROXY.DEFAULTS profile in the FACILITY class

It is reasonable for domain APIs to have access to the domain's LDAP URL and binding information because only:

- The LDAP administrator can create a domain
- A limited number of users have the authority to change, delete or list domain information

The eimRetrieveConfiguration API can also retrieve configuration information, from a specific profile or the current system settings. For more information see "eimRetrieveConfiguration" on page 377.

Tip: Applications, servers, or operating systems can use other APIs (such as registry, identifier, association, access control, and lookup APIs) that should obtain the domain's LDAP URL and binding information from a source the security administrator controls.

Determining why a mapping is not returned

If your application is up and running, it should be able to connect to the EIM domain controller. However, if it does not return expected results for the EIM API, the following could be happening:

- The EIM information you are trying to retrieve is not defined in the EIM domain
- The end user does not have the correct level of authority to the information

Tip: Some things the EIM administrator can consider for investigation:

1. List the EIM information to verify that the EIM information you are looking for is defined.
2. Verify that the user you are connecting with has the required level of authority for the API you are using.

Chapter 8. Messages

Enterprise Identity Mapping (EIM) on z/OS uses message catalogs to store error strings and messages. The error strings explain why a particular return code (or errno) is returned by an EIM API. The messages are issued by the eimadmin utility. Message catalogs makes it easier for software to provide versions of error strings and messages in languages other than English.

All of the messages in this section with the exception of the ITY4*xxx* messages are error strings. The error strings are in the format that is returned by a **catgets()** function.

An error string or message has the following format:

`ITY`*`nnnn`* `text`

nnnn
The message ID number in the message catalog.

The text of an error string might contain an XPG4 conversion specification for a substitution value. A conversion specification has the following format:

`%n$x`

% The start of the conversion specification

n The *n*th argument after the format-string of an fprintf, sprintf, or printf function

$ A delimiter

x The kind of variable (for example, s=string)

More details on XPG4 conversion specifications and their use can be found in *z/OS XL C/C++ Run-Time Library Reference*.

The error message IDs in the EIM message catalog are divided into ranges based by function:

ITY0*xxx*
Error strings that an EIM API returns (across iSeries, zSeries, pSeries, and xSeries platforms)

ITY3*xxx*
Identity cache error messages

ITY4*xxx*
Messages that the eimadmin utility issues

ITY6*xxx*
z/OS-specific error strings

The application programer has two options for handling error strings in an EIM application:

- Retrieve the error string from the message catalog, format the error string into a message, and print the message to the screen or error log.
- Use the eimErr2String API, which retrieves the error string and formats it into a message that can be printed using one of the C or C++ print functions. Note that this only works with EIM error messages; java error messages, such as those from identity cache, are excluded from this function.

An application that works directly with the message catalog needs to do the following:

1. Open the message catalog using the **catopen** function.
2. Read the error string from the message catalog using the **catgets** function.
3. Format the message and fill in any substitution values EIM returns by using one of the *fprintf* family of functions— *fprintf()*, *sprintf()*, *printf()*.
4. Close the message catalog using the catclose function

These functions require the message catalog set number and message ID, which are contained in the EimRC return code parameter on the EIM APIs. See the *z/OS XL C/C++ Run-Time Library Reference* for details on how to use these functions.

The eimErr2String API simplifies the task of creating the message by performing this processing for you.

ITY0001 Insufficient access to EIM data.

Symbolic Identifier (value): EIMERR_ACCESS (1)

Explanation: The bind distinguished name did not have sufficient authority to access the desired EIM data. LDAP returned LDAP_INSUFFICIENT_ACCESS to the requested operation.

Programmer response: Verify the bind distinguished name is a member of the EIM access control group required for the API. The bind distinguished name can be obtained from one of four places:

1. It can be specified on a call to the eimConnect or eimConnectToMaster API.
2. If it is not specified on the API, it can be retrieved from the LDAPBIND class profile that is associated with the caller's user ID.
3. If it is neither specified on the API nor retrieved from the LDABIND class profile associated with the caller, it can be retrieved from the IRR.EIM.DEFAULTS profile in the LDAPBIND class
4. If it is in none of the above places it can be in the IRR.PROXY.DEFAULTS profile in the FACILITY class.

System action: The called function fails.

ITY0002 Access type is not valid.

Symbolic Identifier (value):
EIMERR_ACCESS_TYPE_INVAL (2)

Explanation: The value specified for the access type parameter is not a valid access type.

Programmer response: Correct the errror and try the service again.

System action: The called function fails.

ITY0003 Access user type is not valid.

Symbolic Identifier (value):
EIMERR_ACCESS_USERTYPE_INVAL (3)

Explanation: The value specified for the user access type parameter is either not a valid access type or not supported on this platform.

Programmer response: Check the documentation for the EIM API and verify the user access type is supported. If the user access type is not supported or an incorrect value was specified for the access type, correct the program and try the service again.

System action: The called function fails.

ITY0004 Association type is not valid.

Symbolic Identifier (value):
EIMERR_ASSOC_TYPE_INVAL (4)

Explanation: The value specified for the association type parameter is not a valid type of association.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0005 Attribute name is not valid.

Symbolic Identifier (value): EIMERR_ATTR_INVAL (5)

Explanation: The value specified for the attribute parameter is either not a valid attribute or not supported on this platform.

Programmer response: Check the documentation for the EIM API and verify the attribute is supported. If the attribute is not supported or an incorrect value was specified for the attribute, correct the program and try the service again.

System action: The called function fails.

ITY0006 Attribute not supported.

Symbolic Identifier (value):
EIMERR_ATTR_NOTSUPP (6)

Explanation: The value specified for the handle

attribute is either not a valid attribute or not supported on this platform.

Programmer response: Check the documentation for the EIM API and verify the attribute is supported. If the attribute is not supported or an incorrect value was specified for the attribute, correct the program and try the service again.

System action: The called function fails.

ITY0008 CCSID is outside of valid range or CCSID is not supported.

Symbolic Identifier (value): EIMERR_CCSID_INVAL (8)

Explanation: Not returned on EIM for z/OS.

Programmer response: None.

System action: None.

ITY0009 This change type is not valid with the requested attribute.

Symbolic Identifier (value): EIMERR_CHGTYPE_INVAL (9)

Explanation: The value specified for the attribute change type is not a valid change type value.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0010 Length of EimConfig is not valid.

Symbolic Identifier (value): EIMERR_CONFIG_SIZE (10)

Explanation: Not returned on EIM for z/OS.

Programmer response: None.

System action: None.

ITY0011 Connection already exists.

Symbolic Identifier (value): EIMERR_CONN (11)

Explanation: An attempt was made to use an EIM handle that already has a connection established with an EIM domain.

Programmer response: Correct the error and try the service again.

System action: The invoked function fails.

ITY0012 Connection type is not supported.

Symbolic Identifier (value): EIMERR_CONN_NOTSUPP (12)

Explanation: The value specified for the connection type is either not a valid connection type or not supported on this platform.

Programmer response: Check the documentation for the EIM API and verify the connection type is supported. If the connection type is not supported or an incorrect value was specified for the attribute, correct the program and try the service again.

System action: The called function fails.

ITY0013 Error occurred when converting data between code pages.

Symbolic Identifier (value): EIMERR_DATA_CONVERSION (13)

Explanation: Not returned on EIM for z/OS.

Programmer response: None.

System action: None.

ITY0014 EIM domain entry already exists in EIM.

Symbolic Identifier (value): EIMERR_DOMAIN_EXISTS (14)

Explanation: The EIM domain distinguished name specified in the *ldapURL* parameter is defined on the LDAP host.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0015 Cannot delete a domain when it has registries or identifiers.

Symbolic Identifier (value): EIMERR_DOMAIN_NOTEMPTY (15)

Explanation: The specified EIM domain could not be deleted because it contains identifiers, registry users, or associations.

Programmer response: Delete the identifiers, registry users, or associations and try the service again.

System action: The called function fails.

ITY0016 Length of EimList is not valid. EimList must be at least 20 bytes in length.

Symbolic Identifier (value): EIMERR_EIMLIST_SIZE (16)

Explanation: The value specified for the length of the EimList structure parameter is fewer than 20 bytes.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0017 EimHandle is not valid.

Symbolic Identifier (value):
EIMERR_HANDLE_INVAL (17)

Explanation: The EIM handle does not contain the expected data and cannot be used with any EIM service.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0018 NameInUseAction is not valid.

Symbolic Identifier (value):
EIMERR_IDACTION_INVAL (18)

Explanation: The value specified for the name in use parameter is not valid.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0019 EIM identifier already exists by this name.

Symbolic Identifier (value):
EIMERR_IDENTIFIER_EXISTS (19)

Explanation: The identifier could not be added because another identifier exists in the domain with the same unique name.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0020 More than one EIM identifier was found that matches the requested identifier name.

Symbolic Identifier (value):
EIMERR_IDNAME_AMBIGUOUS (20)

Explanation: The eimListIdentifier or eimRemoveIdentifier service found more than one entry that matches the specified identifier name. This can occur when the non-unique name is used for the name parameter.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0021 A restricted character was used in the object name.

Symbolic Identifier (value): EIMERR_CHAR_INVAL (21)

Explanation: The EIM API detected one of the following characters in the name:

```
, = + > < # ; \ *
```

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY00022 The protect parameter in EimSimpleConnectInfo is not valid.

Symbolic Identifier (value):
EIMERR_PROTECT_INVAL (22)

Explanation: The value specified for the password protection type is either not a valid protection type or not supported on this platform.

Programmer response: Check the documentation for the EIM API and verify the password protection type is supported. If the password protection type is not supported or an incorrect value was specified, correct the program and try the service again.

System action: The called function fails.

ITY0023 Unexpected LDAP error. %1$s

Symbolic Identifier (value): EIMERR_LDAP_ERR (23)

Explanation: An unexpected LDAP error occurred. The *eimrc* parameter contains the name of the LDAP service that returned an error and additional diagnostic information from the LDAP client and from the LDAP server, if available. The substitution text is:

```
ldap client API name - ldap error code:
additional error information
```

Programmer response: Check the LDAP client publication for information on the failing LDAP service and the returned error values. Also check the EIM API documentation for information on the service being performed. Correct the problem and try the service again.

System action: The called function fails.

ITY0024 EIM domain not found or insufficient access to EIM data.

Symbolic Identifier (value): EIMERR_NODOMAIN (24)

Explanation: The domain name in the *ldapURL* parameter does not exist or the bind distinguished name does not have authority to access the EIM data.

Programmer response: Verify the bind distinguished name is a member of an EIM access control group that the EIM API requires. Verify the domain exists in the LDAP directory service. Correct the problem and try the service again.

System action: The called function fails.

ITY0025 EIM identifier not found or insufficient access to EIM data.

Symbolic Identifier (value): EIMERR_NOIDENTIFIER (25)

Explanation: The EIM identifier does not exist or the bind distinguished name used to establish a connection with the EIM domain does not have authority to access the EIM data.

Programmer response: Verify the bind distinguished name is a member of an EIM access control group that the EIM API requires. Verify the identifier exists in the LDAP directory service. Correct the problem and try the service again.

System action: The called function fails.

ITY0026 Unable to allocate internal system object.

Symbolic Identifier (value): EIMERR_NOLOCK (26)

Explanation: Not returned by EIM on z/OS

Programmer response: None.

System action: None.

ITY0027 No memory available. Unable to allocate required space.

Symbolic Identifier (value): EIMERR_NOMEM (27)

Explanation: The EIM API was unable to memory allocate (malloc) storage.

Programmer response: Isolate the reason why the program ran out of storage, correct the problem and try the service again.

System action: The called function fails.

ITY0028 EIM registry not found or insufficient access to EIM data.

Symbolic Identifier (value): EIMERR_NOREG (28)

Explanation: The EIM API could not find the specified registry in the domain or the bind distinguished name did not have access to the registry.

Programmer response: Verify the correct registry name and domain is specified. Verify the bind distinguished name is a member of an access control group that the EIM API requires. Correct the problem and try the service again.

System action: The called function fails.

ITY0029 Registry user not found or insufficient access to EIM data.

Symbolic Identifier (value): EIMERR_NOREGUSER (29)

Explanation: The EIM API could not find the specified registry user in the domain or the bind distinguished name did not have access to the registry.

Programmer response: Verify the correct registry user, registry name and domain is specified. Verify the bind distinguished name is a member of an access control group that an EIM API requires. Correct the problem and try the service again.

System action: The called function fails.

ITY0030 EIM environment is not configured.

Symbolic Identifier (value): EIMERR_NOTCONFIG (30)

Explanation: The EIM API could not find the *ldapURL* or the registry name in a RACF profile.

Programmer response: If the EIM API requires an *ldapURL* and the application is using EIM configuration information associated with the caller's user profile, verify that the EIM segment for the user profile has the name of a profile in the LDAPBIND class. Verify the LDAPBIND class has a host name in the LDAPHOST field of the PROXY segment and a domain distinguished name (DN) in the DOMAINDN field of the EIM segment. If the application is using the system defaults from the IRR.EIM.DEFAULTS LDAPBIND clss profile or the IRR.PROXY.DEFAULTS FACILITY class profile, verify the LDAP host name and EIM domain distinguished name are defined. If the EIM API requires the system default registry name, then verify the IRR.PROXY.DEFAULTS FACILITY class profile contains a registry name in the LOCALREG field of the EIM segment. Correct the problem and try the service again.

System action: The called function fails.

ITY0031 Not connected to LDAP.

Symbolic Identifier (value): EIMERR_NOT_CONN (31)

Explanation: The EIM API requires an EIM handle that is connected to an EIM domain.

Programmer response: Issue an eimConnect or eimConnect service for the EIM handle and try the service again.

System action: The called function fails.

ITY0032 The system is not configured to connect to a secure port. Connection type of EIM_CLIENT_AUTHENTICATION is not valid.

Symbolic Identifier (value): EIMERR_NOT_SECURE(32)

Explanation: The URL for the EIM domain controller does not begin with ldaps.

Programmer response: Verify the URL is correct and that the correct option is specified on the EIM API, then try the service again.

System action: The called function fails.

ITY0033 System registry not found.

Symbolic Identifier (value): EIMERR_NO_SYSREG (33)

Explanation: The eimAddApplicationRegistry API requires the name of a system registry. The registry does not exist in the EIM domain or the bind distinguished name (DN) is not a member of one of the access control groups that the eimAddApplicationRegistry API requires.

Programmer response: Verify the correct system registry name is provided. Verify the bind disiinguished name is a member of one of the access control groups that the eimAddApplication registry requires. Correct the problem and try the service again.

System action: The called function fails.

ITY0034 Missing required parameter.

Symbolic Identifier (value): EIMERR_PARM_REQ (34)

Explanation: A required parameter for the EIM API is missing.

Programmer response: Check the EIM API documentation, identify the missing parameter, correct the problem, and try the service again.

System action: The called function fails.

ITY0035 Pointer parameter is not valid.

Symbolic Identifier (value): EIMERR_PTR_INVAL (35)

Explanation: Not returned by EIM on z/OS

Programmer response: None.

System action: None.

ITY0036 LDAP connection is for read only.

Symbolic Identifier (value): EIMERR_READ_ONLY (36)

Explanation: The EIM API tried to add, delete, or change information in an EIM domain, but the EIM handle is connected to an LDAP server that is read only.

Programmer response: Create a new handle that is connected to a master LDAP server and try the service again. .

System action: The invoked function fails

ITY0037 Registry entry already exists in EIM.

Symbolic Identifier (value): EIMERR_REGISTRY_EXISTS (37)

Explanation: The EIM API tried to add a system or application registry to an EIM domain and a registry with the same name is defined in the domain.

Programmer response: Create a new handle that is connected to a master LDAP server and try the service again.

System action: The called function fails.

ITY0038 Requested registry kind is not valid.

Symbolic Identifier (value): EIMERR_REGKIND_INVAL (38)

Explanation: The value specified for the registry kind is not a valid registry kind.

Programmer response: Check the documentation for the EIM API, correct the problem, and try the service again.

System action: The called function fails.

ITY0039 Local registry name is too large.

Symbolic Identifier (value): EIMERR_REGNAME_SIZE (39)

Explanation: This is not returned by EIM on z/OS.

Programmer response: None.

System action: None.

ITY0040 Cannot delete a registry when an application registry has this system registry defined.

Symbolic Identifier (value): EIMERR_REG_NOTEMPTY (40)

Explanation: The specified EIM registry could not be deleted because an application registry is defined for this system registry.

Programmer response: Delete the application registry and try the service again.

System action: The called function fails.

ITY0041 Unexpected error accessing parameter.

Symbolic Identifier (value): EIMERR_SPACE (41)

Explanation: This is not returned by EIM on z/OS.

Programmer response: None.

System action: None.

ITY0042 EimSSLInfo is required.

Symbolic Identifier (value): EIMERR_SSL_REQ (42)

Explanation: The EIM domain controller URL begins with ldaps, but the SSL information was not specified as a parameter to the EIM API.

Programmer response: Correct the EIM domain controller URL or parameter list for the EIM API and try the service again.

System action: The called function fails.

ITY0043 Length of unique name is not valid.

Symbolic Identifier (value): EIMERR_UNIQUE_SIZE (43)

Explanation: The length of the uniqueName is not 20 bytes longer than the length of the identifier.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0044 Unknown exception or unknown system state.

Symbolic Identifier (value): EIMERR_UNKNOWN (44)

Explanation: Not returned by EIM on z/OS

Programmer response: None.

System action: None.

ITY0045 URL has no distinguished name (required).

Symbolic Identifier (value): EIMERR_URL_NODN (45)

Explanation: The value specified for the *ldapURL* parameter does not contain a distinguished name.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0046 URL has no domain (required).

Symbolic Identifier (value):
EIMERR_URL_NODOMAIN (46)

Explanation: The distinguished name portion of the *ldapURL* parameter does not begin with `ibm-eimDomainName=` or `ibm-eimdomainname=`.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0047 URL does not have a host.

Symbolic Identifier (value): EIMERR_URL_NOHOST (47)

Explanation: The value specified for the *ldapURL* parameter does not contain an LDAP host name.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0048 URL has no port (required).

Symbolic Identifier (value): EIMERR_URL_NOPORT (48)

Explanation: Not returned by EIM on z/OS

Programmer response: None.

System action: None.

ITY0049 URL does not begin with ldap:// or ldaps://.

Symbolic Identifier (value):
EIMERR_URL_NOTLDAP (49)

Explanation: The value specified for the *ldapURL* parameter does not begin with `ldap://` or `ldaps://`.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0050 LDAP connection can only be made to a replica LDAP server.

Symbolic Identifier (value):
EIMERR_URL_READ_ONLY (50)

Explanation: The EIM API requires a connection to a master or writable server.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0051 Configuration URL is too large.

Symbolic Identifier (value): EIMERR_URL_SIZE (51)

Explanation: Not returned by EIM on z/OS

Programmer response: None.

System action: None.

ITY0052 The EimIdType value is not valid.

Symbolic Identifier (value):
EIMERR_IDNAME_TYPE_INVAL (52)

Explanation: The value specified for the type of identifier is not one of the allowed values.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0053 Length of EimAttribute is not valid.

Symbolic Identifier (value): EIMERR_ATTRIB_SIZE (53)

Explanation: The length of the value for the handle attribute is fewer than 8 bytes.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0054 Connection type is not valid.

Symbolic Identifier (value): EIMERR_CONN_INVAL (54)

Explanation: The value specified for the connection type is either not a correct connection time or not supported on this platform.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0055 Registry name must be NULL when access type is not EIM_ACCESS_REGISTRY.

Symbolic Identifier (value):
EIMERR_REG_MUST_BE_NULL (55)

Explanation: The value specified for the access type requires the registry name parameter to be NULL.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0056 Unexpected object violation.

Symbolic Identifier (value):
EIMERR_UNEXP_OBJ_VIOLATION (56)

Explanation: The EIM API attempted to retrieve the entry UUID attribute for an LDAP entry and it was not returned.

Programmer response: Contact your LDAP administrator or system programmer. Provide this person with the name of the EIM API and the error number (errno) and error string.

System action: The called function fails.

ITY0057 Reserved field is not valid.

Symbolic Identifier (value):
EIMERR_RESERVE_INVAL (57)

Explanation: Not returned by EIM on z/OS

Programmer response: None.

System action: None.

ITY0058 Credentials must be NULL for the specified connection type.

Symbolic Identifier (value):
EIMERR_CREDS_MUST_BE_NULL (58)

Explanation: The connection info parameter of the EIM API does not have a NULL value for the creds field in the connection info structure.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0059 Error occurred after the domain object was created.

Symbolic Identifier (value):
EIMERR_DOMAIN_UNUSABLE (59)

Explanation: The EIM API was unable to create the groups, identifier, source mappings, or registries containers in the EIM domain. The domain is in an unusable state. The return code is the value returned from an ldap_add_s service.

Programmer response: Contact your LDAP administrator or your systems programmer. Provide this person with the name of the EIM API, the return code, and the error string. The LDAP administrator will need to delete the domain.

System action: The called function fails.

ITY0060 Policy filter type is not valid.

Symbolic Identifier (value):
EIMERR_POLICY_FILTER_TYPE_INVAL (60)

Explanation: The value specified for the policy filter type parameter is not a valid policy filter type.

Programmer response: Refer to the API documentation for valid policy filter types, correct the error and try the service again.

System action: The called function fails.

ITY0061 Policy filter value not found for the specified registry.

Symbolic Identifier (value):
EIMERR_NOPOLICYFILTER (61)

Explanation: The filter value is not defined for the source registry.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0062 Registry type is not valid.

Symbolic Identifier (value):
EIMERR_REGTYPE_INVAL(62)

Explanation: The value specified for the registry type is not one of the supported values.

Programmer response: Refer to the API documentation for valid registry types, correct the error, and try the service again.

System action: The called function fails.

ITY0063 User identity type is not valid.

Symbolic Identifier (value):
EIMERR_USER_IDENTITY_TYPE_INVAL (63)

Explanation: The value specified for the user identity type parameter is not a valid type.

Programmer response: Refer to the API documentation for valid user identity types, correct the error, and try the service again.

System action: The called function fails.

ITY0064 Length of EimUserIdentity is not valid.

Symbolic Identifier (value):
EIMERR_USER_IDENTITY_SIZE (64)

Explanation: The value specified for the length of the user identity information structure parameter is not sufficient. The minimum storage required for the user identity information structure is 16 bytes.

Programmer response: Obtain a larger block of storage for the user identity information structure and update the length value with the new size. Then try the eimFormatUserIdentity service again.

System action: The called function fails.

ITY0065 User identity format type is not valid.

Symbolic Identifier (value):
EIMERR_USER_IDENTITY_FORMAT_TYPE_INVAL (65)

Explanation: The value specified for the format type parameter is not a valid type.

Programmer response: Change the user identity format type parameter to a supported value, and try the service again.

System action: The called function fails.

ITY0066 Distinguished Name (DN) is not valid.

Symbolic Identifier (value): EIMERR_INVALID_DN (66)

Explanation: An error occurred while attempting to parse a distinguished name. The DN may have been specified as a parameter or extracted from a certificate parameter.

Programmer response: Correct the parameter in error and try the service again.

System action: The called function fails.

ITY0067 Certificate data is not valid.

Symbolic Identifier (value):
EIMERR_CERTIFICATE_INVAL(67)

Explanation: An error occurred while decoding or parsing the certificate parameter.

Programmer response: Programmer Response: Ensure the certificate parameter is a valid base64 or DER encoded X.509 certificate.

System action: The called function fails.

ITY0068 Configuration format is not valid.

Symbolic Identifier (value):
EIMERR_CONFIG_FORMAT_INVAL (68)

Explanation: The value specified in the configuration info parameter for the configuration format is not a valid type.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0069 The specified type is not valid.

Symbolic Identifier (value): EIMERR_TYPE_INVAL (69)

Explanation: The value specified for the type parameter is not a valid.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY0070 The specified function is not supported by the EIM version.

Symbolic Identifier (value): EIMERR_FUNCTION_NOT_SUPPORTED (70)

Explanation: The application called a functioin that requires that the EIM domain controller (the LDAP Server) have new schema elements that were not found on the connected domain controller.

Programmer response: Use an EIM domain controller that has the requisite support, or upgrade the existing EIM domain controller to the minimum requisite level.

System action: The called function fails.

ITY0071 Unable to find LDAP schema.

Symbolic Identifier (value): EIMERR_LDAP_SCHEMA_NOT_FOUND (71)

Explanation: An attempt to obtain the subschemasubentry suffix failed for the specified, configured, or connected EIM domain controller.

Programmer response: Specify, configure, or connect to an EIM domain controller that is supported by EIM.

System action: The called function fails.

ITY3300E Authentication Context exception. Length not correct for ″{0}″. Length found: {1}. Available length: {2}.

Explanation: A length inconsistency was detected in an internal identity context data structure.

User response: Ensure the specified identity context is valid. If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY3301E Authentication Context exception. The specified array is not large enough to contain identity context headers.

Explanation: An error was detected while parsing an identity context. The length of the specified array is not large enough to hold a valid identity context.

User response: Ensure the specified identity context is valid. If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY3302E Authentication Context exception. The value of the ″length″ field {0} does not match the length of the array {1}.

Explanation: A length inconsistency was detected in an internal identity context data structure.

User response: Ensure the specified identity context is valid. If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY3303E Authentication Context exception. OID not correct for Type 1 authentication context.

Explanation: The OID from the specified identity context does not match the Type 1 authentication context OID. The method called can only process identity contexts which are Type 1 authentication contexts.

User response: Ensure identity context is a Type 1 authentication context and retry the method.

System action: Processing stops.

ITY3304E Authentication Context exception. OID in authentication context does not match Type 1 OID.

Explanation: The internal OID from the specified identity context does not match the Type 1 authentication context OID. The method called can only process identity contexts which are Type 1 authentication contexts.

User response: Ensure identity context is a Type 1 authentication context and retry the API.

System action: Processing stops.

ITY3350E Identity Context exception. Identity Context not found.

Explanation: An identity context could not be retrieved for the specified context credentials. The context credentials may not have been valid or may have timed out.

User response: Ensure the context credentials are valid and are used before they time out.

System action: Processing stops.

ITY3351E **Identity Context exception. Identity Context damaged.**

Explanation: Not returned on z/OS.

User response: None.

System action: None.

ITY3352E **Identity Context exception. Error in API: {0}. Exception from EIM. EIM Exception: {1}.**

Explanation: Not returned on z/OS.

User response: None.

System action: None.

ITY3353E **Missing required parameter. ″{0}″**

Explanation: A required parameter was not supplied.

User response: Supply the required parameter and retry the method.

System action: Processing stops.

ITY3354E **Illegal argument for parameter: ″{0}″**

Explanation: A parameter contains an illegal or inappropriate argument.

User response: Consult the Javadoc for the correct parameter syntax. Correct the illegal parameter value and retry the method.

System action: Processing stops.

ITY3355E **Input class not expected class type. Expected class: ″{0}″**

Explanation: A parameter is not of the expected class type. The returned exception will include a substitution for the expected class.

User response: Ensure all parameters match the method signature. Use expected class to resolve the problem and try the method again.

System action: Processing stops.

ITY4001 **Error** *errno* **returned from attempt to open message catalog** *file* **--** *errnoText*

Explanation: The utility failed to open the message catalog named *file*. The open attempt sets the error code *Errno*. *errnoText* is the associated explanation.

User response: Possible causes for this message include an incomplete NLSPATH definition or insufficient access to the file. Correct the problem and restart the utility.

System action: The utility fails.

ITY4002 **Error** *errno* **returned from attempt to retrieve message from catalog** *file* **--** *errnoText*

Explanation: The utility failed to read a message from the message catalog named *file*. The read attempt sets the error code *Errno*. *errnoText* is the associated explanation.

User response: Possible causes for this message include a corrupt catalog file or a file having a different service level than the utility. Correct the problem and restart the utility.

System action: The utility fails.

ITY4010 **No argument value specified for option** *character***.**

Explanation: The option *character* specified requires an argument value, but none was specified.

User response: Restart the utility specifying a value for the indicated option, or omit the option altogether.

System action: The utility fails.

ITY4011 **Option** *character* **not recognized. Specify '-?' for utility syntax.**

Explanation: The *character* specified is not a defined option.

User response: You can review utility syntax by specifying the -? option. Restart the utility specifying correct option characters.

System action: The utility fails.

ITY4012 *option* **not specified.**

Explanation: The *option* names the entity that is required for the function but was not specified through a command line option or input data record.

User response: Restart the utility, making sure to include a value for the requested option.

System action: The utility fails.

ITY4013 **Specified** *type* **value not supported --** *value*

Explanation: *Type* describes the entity for which an unsupported *value* was specified.

User response: Refer to the utility documentation for allowable entity values. Restart the utility, specifying an allowable value.

System action: The utility fails.

ITY4014 **Specified combination of action '*character*' and object type '*character*' not supported.**

Explanation: The utility does not offer any function corresponding to the specified action and object option combination as indicated by *character*.

User response: Choose another option combination and restart the utility.

System action: The utility fails.

ITY4015 **Please enter LDAP bind password for *bindDN*:**

Explanation: The utility prompts for an LDAP bind password if not specified as a command line option. The password should be the one associated with the LDAP *bindDN* specified.

User response: Enter the password as requested. The value will not be displayed.

System action: The utility allows the user one chance to input a non-NULL password value. If one is not specified, the utility fails.

ITY4016 **Please enter key file password for *fileName*:**

Explanation: The utility prompts for an SSL key database file password if the specified file exists but its password was not specified as a command line option. The password should be the one associated with the *fileName* specified. Alternatively, you can specify an SSL password stash file by prefixing the stash file name with ″file://″.

User response: Enter the password as requested. The value will not be displayed.

System action: The utility allows the user one chance to input a non-NULL password value. If one is not specified, the utility fails.

ITY4017 **Domain DN must be a distinguished name beginning with 'ibm-eimDomainName='.**

Explanation: The value specified is incorrect or incomplete.

User response: Restart the utility, making sure the domain value is a distinguished name beginning with `'ibm-eimDomainName='`.

System action: The utility fails.

ITY4020 **eimadmin (*version*) started *time* with options *commandLine***

Explanation: The utility issues this informational message when beginning its processing of input records. *Version* indicates the program level. *Time* indicates the date and time that processing began. *CommandLine* is an approximation of the string issued to start the utility.

User response: User Response:

None.

System action: Processing continues.

ITY4021 **Processing ended normally.**

Explanation: The utility processed all records from the input file; however, errors might have occurred along the way.

User response: Check the preceeding ITY4022 message to learn if any errors occurred. If so, correct them and, if appropriate, restart the utility against a file of the previously-failing records.

System action: The utility stops.

ITY4022 *Count* **records processed -- *successCount* successful; *failCount* failed.**

Explanation: When processing an input file, the utility issues this message as a progress indicator one time every 50 records and as a completion summary statement. *Count* is the number of data lines processed from the input file. *SuccessCount* is the number processed without error, while *failCount* indicates the number of records for which errors occurred. The *count* value at the end of processing should equal the number of data lines in the input file if message ITY4021 is issued.

User response: If *failCount* is greater than zero, errors occurred. Review preceding error messages to determine where and why errors occurred. Correct the errors and, if appropriate, restart the utility against a file of the previously-failing records.

System action: The utility continues if there are remaining unprocessed input records unless a severe error has occurred.

ITY4023 **Processing stopped due to error.**

Explanation: The utility stopped processing before reaching the end of data records in the input file because it encountered a severe error.

User response: The last data record error message should identify the problem, but less severe errors might have occurred as well. Review the error messages to determine where and why errors occurred. Correct the

errors and, if appropriate, restart the utility against a file of the previously-failing records.

System action: The utility stops

ITY4024 Label definition line not found in input file.

Explanation: The utility reached the end of the input file without identifying the label definition line.

User response: Verify the input file specified is the one intended and that it has a label line that is not prefixed with a comment character. Restart the utility, specifying an input file with proper syntax.

System action: The utility stops.

ITY4025 Unrecognized label found starting at column *position* **on input line** *number***.**

Explanation: The utility detected characters in the first non-blank, non-comment line that do not constitute a supported label. *Position* is the character offset from the beginning of the line, identified by *number*.

User response: Verify the input file specified is the one intended and that the label line contains only supported labels. Restart the utility, specifying an input file with proper syntax.

System action: The utility stops.

ITY4026 Missing final label delimiter '*character***' on input line** *number***.**

Explanation: The utility did not find the required field delimeter, *character*, at the end of the label line, which is line *number* of the input file.

User response: Insert the missing delimiter and restart the utility.

System action: The utility stops.

ITY4027 Length of input line *number* **exceeds limit characters.**

Explanation: The length of input file line *number* is greater than the allowed *limit*.

User response: Shorten the data lines that exceed the limit, and restart the utility.

System action: The utility stops.

ITY4028 Error occurred while processing input line *number***.**

Explanation: The utility encountered an error while processing line *number* from the input file. The next line of error ouput echoes the data line from the input file.

User response: Refer to the error message immediately preceding this message to discover the cause of the error. Correct the error, and restart the utility.

System action: The utility stops if a severe error occurred; otherwise it continues processing data records.

ITY4030 Service *name* **returned error** *code* **--** *text*

Explanation: The utility called a service, identified by *name*, that returned an error. *Code* is the error number, and *text* is the associated error message.

User response: The error text should indicate the cause of the problem. If not, refer to error documentation for the service. Correct the error, and restart the utility.

System action: The utility stops if a severe error occurred; otherwise it continues processing data records.

ITY4031 Service *name* **returned error** *code* **--** *reason***.**

Explanation: The utility called a service, identified by *name*, that returned an error. *Code* is the error number, and *reason* is the corresponding reason number. This message is issued in place of ITY4030 when the text for the *reason* could not be found. This can happen for an EIM service error if eimErr2String() encounters an error reading its message catalog.

User response: The error codes should indicate the cause of the problem. Correct the error and restart the utility. Investigate the problem with the EIM message catalog.

System action: The utility stops if a severe error occurred, otherwise it continues processing data records.

ITY4040 Internal error occurred -- *text*

Explanation: An unexpected internal error occurred as described by *text*.

User response: If the problem re-occurs and cannot be solved, contact service.

System action: The utility stops if a severe error occurred, otherwise it continues processing data records.

ITY4041 Program exception occurred.

Explanation: An unexpected program exception stopped utility processing.

User response: Review the generated CEEDUMP for diagnostic information that can help you resolve the problem. It is unlikely that the requested function completed successfully. List the entity specified for the function to determine its status and retry the function if

necessary. If the exception occurrred while the utility was processing records from an input file, message ITY4028 indicates the input line number and message ITY4022 indicates the number of records successfully processed.

System action: Processing stops. The recovery routine generates a symptom record.

ITY4042 Specified combination of option option1 and option2 is not supported.

Explanation: The utility failed during validation of the specified options. The options option1 and option2 may not be specified in conjunction with each other.

Programmer response: Correct the problem by removing one of the options and try the command again.

System action: The utility stops if a severe error occurred; otherwise it continues processing data records.

ITY4043 Could not find the certificate file -- service -- error text

Explanation: The utility performed validation of the certificate file specified which failed. The indicated service did not succeed and the problem encountered is described in the included error text. The specified certificate file may not exist, may be a zero length file, or may not be a regular file or symbolic link to a regular file.

Programmer response: The error text in the message should indicate the problem. If not, refer to error documentation for the service. Correct the error and restart the utility.

System action: The utility stops if a severe error occurred; otherwise it continues processing data records.

ITY4044 Could not open the certificate file

Explanation: The certificate file specified could not be opened because file may not be a regular file or fopen() failed.

Programmer response: Correct the error and run the utility command again.

System action: The utility fails.

ITY4045 Could not read the certificate file -- service -- error text

Explanation: The utility could not read the certificate file specified. The indicated service did not succeed and the problem encountered is described in the included error text.

Programmer response: The error text in the message should indicate the problem. If not, refer to error documentation for the service. Correct the error and restart the utility.

System action: The utility stops if a severe error occurred; otherwise it continues processing data records.

ITY4046 Expected string ″-----END CERTIFICATE-----″ not found

Explanation: The certificate file name passed to the utility encountered a base64 decoding problem. The ″-----BEGIN CERTIFICATE-----″ text string was found but the corresponding ″-----END CERTIFICATE-----″ text string was not found in the certificate file.

Programmer response: Specify a valid certificate file and run the utility command again.

System action: The utility stops if a severe error occurred; otherwise it continues processing data records.

ITY6002 RACROUTE REQUEST=EXTRACT error retrieving EIM configuration information from the callers's USER profile. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_USER_XTR (6002)

Explanation: The EIM API failed while retrieving EIM information from RACF. A RACROUTE REQUEST=EXTRACT error occurred while retrieving the EIM segment from the caller's USER profile. Failing user ID and return codes appear in the EimRC substitution text. The substition text is:

```
USER(user id ) SAF RC(xxxxxxxx) RACF RC(xxxxxxxx)
RACF RSN(xxxxxxxx)
```

The return and reason codes are in hex. The RACROUTE return codes are documented in *z/OS Security Server RACROUTE Macro Reference*.

Programmer response: Use the return codes to resolve the problem in the user EIM segment and try the service again.

System action: The called function fails.

ITY6003 RACROUTE REQUEST=EXTRACT error retrieving EIM information from a RACF profile. %1$s

Symbolic Identifier (value): EIMERR_ZOS_XTR_EIM (6003)

Explanation: The EIM API failed while retrieving EIM information from RACF. A RACROUTE REQUEST=EXTRACT error occurred while retrieving the EIM segment from a RACF profile. Failing user ID, class, profile name and return codes appear in the EimRC substitution text. The substitution text is:

```
USER(user ID) CLASS(class) PROFILE(profile name)
SAF RC(xxxxxxxx) RACF RC(xxxxxxxx) RACF
RSN(xxxxxxxx)
```

The return and reason codes are in hex. The RACROUTE return codes are documented in *z/OS Security Server RACROUTE Macro Reference*.

Programmer response: Use the return codes to resolve the problem in the profile and try the service again.

System action: The called function fails.

ITY6004 EIM domain distinguished name is missing. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_XTR_DOMAINDN (6004)

Explanation: The EIM API failed while retrieving EIM information from RACF. The EIM segment DOMAINDN field has a length of zero. Failing user ID, class and profile name appear in the EimRC substitution text. The substitution text is:

```
USER(user id) CLASS(class) PROFILE(profile name)
```

Programmer response: Ensure the EIM segment DOMAINDN field is defined properly and try the service again.

System action: The called function fails.

ITY6005 RACROUTE REQUEST=EXTRACT error retrieving PROXY information from a RACF profile. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_XTR_PROXY (6005)

Explanation: The EIM API failed while retrieving PROXY information from RACF. A RACROUTE REQUEST=EXTRACT error occurred while retrieving the PROXY segment from a RACF profile. Failing user ID, class, profile name and return codes will appear in the EimRC substitution text. The substitution text is:

```
USER(user id) CLASS(class) PROFILE(profile name)
SAF RC(xxxxxxxx) RACF RC(xxxxxxxx) RACF
RSN(xxxxxxxx)
```

The return and reason codes are in hex. The RACROUTE return codes are documented in *z/OS Security Server RACROUTE Macro Reference*.

Programmer response: Use the return codes to resolve the problem in the profile and try the service again.

System action: The called function fails.

ITY6006 PROXY LDAP host is missing. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_XTR_LDAPHOST (6006)

Explanation: The EIM API failed while retrieving PROXY information from RACF. The PROXY segment LDAPHOST field has a length of zero. Failing user ID, class and profile name appear in the EimRC substitution text. The substitution text is:

```
USER(user id) CLASS(class) PROFILE(profile name)
```

Programmer response: Ensure the PROXY segment LDAPHOST field is defined properly and try the service again.

System action: The called function fails.

ITY6007 PROXY bind distinguished name is missing. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_XTR_BINDDN (6007)

Explanation: The EIM API failed while retrieving PROXY information from RACF. The PROXY segment BINDDN field has a length of zero. Failing user ID, class and profile name appear in the EimRC substitution text. The substitution text is:

```
USER(user id) CLASS(class) PROFILE(profile name)
```

Programmer response: Ensure the PROXY segment BINDDN field is defined properly and try the service again.

System action: The called function fails.

ITY6008 R_DCEKEY callable service failed. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_R_DCEKEY (6008)

Explanation: The EIM API failed while retrieving PROXY information from RACF. An error occurred during R_DCEKEY callable service processing. Failing user ID, class, profile name and return codes appear in the EimRC substitution text. The substitution text is:

```
USER(user id) CLASS(class) PROFILE(profile name)
SAF RC(xxxxxxxx) RACF RC(xxxxxxxx) RACF
RSN(xxxxxxxx)
```

The return and reason codes are in hex. The R_DCEKEY return codes are documented in *z/OS Security Server RACF Callable Services*.

Programmer response: Use the return codes to resolve the R_DCEKEY problem and try the service again.

System action: The called function fails.

ITY6009 R_DCEKEY callable service failed. Bind password is missing. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_R_DCEKEY_BINDPW (6009)

Explanation: The EIM API failed while retrieving PROXY information from RACF. An error occurred during R_DCEKEY callable service processing. The PROXY segment BINDPW field has a length of zero. Failing user ID, class, profile name and return codes appear in the EimRC substitution text. The substitution text is:

```
USER(user id) CLASS(class) PROFILE(profile name)
SAF RC(xxxxxxxx) RACF RC(xxxxxxxx) RACF
RSN(xxxxxxxx)
```

The return and reason codes are in hex. The R_DCEKEY return codes are documented in *z/OS Security Server RACF Callable Services*.

Programmer response: Ensure the PROXY segment BINDPW field is defined properly and try the service again.

System action: The called function fails.

ITY6010 No task or address space ACEE was found.

Symbolic Identifier (value):
EIMERR_ZOS_NO_ACEE (6010)

Explanation: The EIM API service failed because a task or address space ACEE could not be found.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY6011 Error occurred when converting data between code pages. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_DATA_CONVERSION (6011)

Explanation: An error occurred when converting data between code pages. Check the EimRC substitution text for more specific code page errors. The EIM API is unable to determine the current code page or cannot translate between the code pages specified in the substitution text. The substition text is:

```
Failed converting to UTF-8 from <current locale
codeset>
```

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY6012 The EIM API is not supported.

Symbolic Identifier (value): EIMERR_API_NOTSUPP (6012)

Explanation: The called function is not available to z/OS programs.

Programmer response: Check the documentation for the EIM API for alternative methods of providing the function. Correct the program and try the service again.

System action: The called function fails.

ITY6013 Password protection value not supported.

Symbolic Identifier (value):
EIMERR_PROTECT_NOTSUPP (6013)

Explanation: The value specified for password protection is either incorrect or not supported on this platform.

Programmer response: Check the documentation for the EIM API and verify the value is supported. If the value is not supported or an incorrect value was specified for the attribute, correct the program and try the service again.

System action: The called function fails.

ITY6014 The specified value for the function parameter is not valid.

Symbolic Identifier (value):
EIMERR_ZOS_FUNCTION_INVAL (6014)

Explanation: The EIM API failed verification of the function parameter. The specified value for the function parameter is not one of the supported values.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY6015 The EIM API failed verification of the user identity parameter. The user identity value must be NULL.

Symbolic Identifier (value):
EIMERR_USERID_MUST_BE_NULL(6015)

Explanation: The user identity parameter must be NULL when specified with IRR.EIM.DEFAULTS or IRR.PROXY.DEFAULTS profiles names.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY6016 The length of the specified profile name is not valid.

Symbolic Identifier (value): EIMERR_PROFILE_SIZE (6016)

Explanation: The EIM API failed verification of the value specified for the profile name. The profile name passed exceeded the maximum length for a profile which is 246 characters.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY6017 The length of the specified user identity is not valid.

Symbolic Identifier (value): EIMERR_USERID_SIZE (6017)

Explanation: The EIM API failed verification of the value specified for the user identity. The value passed for the user identity exceeded the maximum length which is 8 characters.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY6018 The bind password is missing in the specified profile. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_XTR_BINDPW (6018)

Explanation: The EIM API failed while retrieving the BINDPW field of the PROXY segment from the specified profile. The PROXY segment BINDPW field has a length of zero. Failing user ID, class and profile name appear in the EimRC substitution text. The substitution text is:

```
USER(user id) CLASS(class) PROFILE(profile name)
```

Programmer response: Ensure the PROXY segment BINDDN field is defined properly and try the service again.

System action: The called function fails.

ITY6019 The length of the specified bind password is not valid.

Symbolic Identifier (value):
EIMERR_ZOS_BINDPW_SIZE (6019)

Explanation: The EIM API failed verification of the value specified for the bind password. The value passed exceeds the maximum length allowed which is 128 characters.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY6020 The length of the specified bind distinguished name is not valid.

Symbolic Identifier (value):
EIMERR_ZOS_BINDDN_SIZE (6020)

Explanation: The EIM API failed verification of the value specified for the bind distinguished name(DN). The value passed exceeds the maximum length allowed which is 1023 characters.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY6021 The RACF command invoked by the R_admin callable service failed - %1$s

Symbolic Identifier (value):
EIMERR_ZOS_RADMIN_CMD_ERROR (6021)

Explanation: The EIM API failed while attempting to create, update, or delete a security product profile with the R_admin callable service. If the calling program passed an EimRC structure pointer to the API, the substitution text will contain the return and reason codes returned from R_admin, the failing command string passed to R_admin, and any error messages returned from the R_admin callable service. The substitution text is:

```
 R_admin command(command string)
SAF RC(return code) RACF RC(return code)
RACF RSN(reason code)
Command error message 1
Command error message 2
...
```

The return and reason codes are in decimal. The R_admin callable service return and reason codes are documented in *z/OS Security Server RACF Callable Services*.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY6022 The invocation of the R_admin callable service failed - %1$s

Symbolic Identifier (value):
EIMERR_ZOS_RADMIN_ERROR (6022)

Explanation: The EIM API failed while attempting to a call to the R_admin callable service. If the calling program passed an EimRC structure pointer to the API, the substitution text will contain the return and reason codes returned from R_admin, the command string passed to R_admin, and any error messages returned from the R_admin callable service. The substitution text is:

```
 R_admin command(command string)
SAF RC(return code) RACF RC(return code)
RACF RSN(reason code)
Command error message 1
Command error message 2
...
```

The return and reason codes are in decimal. The R_admin callable service return and reason codes are documented in *z/OS Security Server RACF Callable Services*.

Programmer response: Correct the error and try the service again.

System action: The called function fails.

ITY6023 The R_GetInfo callable service failed. Not authorized to use this service. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_RGETINFO_AUTH (6023)

Explanation: The EIM API failed while retrieving EIM information from RACF. The R_GetInfo callable service returned an authorization failure. Failing user ID, class, profile name and return codes appear in the EimRC substitution text. The substitution text is:

```
USER(user ID) CLASS(class) PROFILE(profile name)
SAF RC(xxxxxxxx) RACF RC(xxxxxxxx) RACF
RSN(xxxxxxxx)
```

The EimRC substitution text can contain null values, such as USER(), for parameters which are not applicable to the failing request. The return and reason codes are in hex. The R_GetInfo return codes are documented in *z/OS Security Server RACF Callable Services*.

Programmer response: Use the return codes to resolve the R_GetInfo problem and try the service again.

System action: The called function fails.

ITY6024 The R_GetInfo callable service failed. Field-level access checking failed. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_RGETINFO_FLAC (6024)

Explanation: The EIM API failed while retrieving EIM information from RACF. The R_GetInfo callable service returned a field-level access check failure. Failing user ID, class, profile name and return codes 365 appear in the EimRC substitution text. The substitution text is:

```
USER(user ID) CLASS(class) PROFILE(profile name)
SAF RC(xxxxxxxx) RACF RC(xxxxxxxx) RACF
RSN(xxxxxxxx)
```

The EimRC substitution text can contain null values, such as USER(), for parameters not applicable to the failing request. The return and reason codes are in hex. The R_GetInfo return codes are documented in *z/OS Security Server RACF Callable Services*.

Programmer response: Use the return codes to resolve the R_GetInfo problem and try the service again.

System action: The called function fails.

ITY6025 The R_GetInfo callable service failed. Extract failed. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_RGETINFO_XTR (6025)

Explanation: The EIM API failed while retrieving EIM information from RACF. The R_GetInfo callable service encountered an unexpected error while attempting to extract EIM information. Failing user ID, class, profile name and return codes appear in the EimRC substitution text. The substitution text is:

```
USER(user ID) CLASS(class) PROFILE(profile name)
SAF RC(xxxxxxxx) RACF RC(xxxxxxxx) RACF
RSN(xxxxxxxx)
```

The EimRC substitution text can contain null values, such as USER(), for parameters not applicable to the failing request. The return and reason codes are in hex. The R_GetInfo return codes are documented in *z/OS Security Server RACF Callable Services*.

Programmer response: Look for related RACF messages, ensure that the EIM information being retrieved was defined properly, and try the service again.

System action: The called function fails.

ITY6026 The R_GetInfo callable service failed. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_RGETINFO_ERROR (6026)

Explanation: The EIM API failed while retrieving EIM information from RACF. The R_GetInfo callable service returned a processing failure. Failing user ID, class, profile name and return codes appear in the EimRC substitution text. The substitution text is:

```
USER(user ID) CLASS(class) PROFILE(profile name)
SAF RC(xxxxxxxx) RACF RC(xxxxxxxx) RACF
RSN(xxxxxxxx)
```

The EimRC substitution text can contain null values, such as USER(), for parameters not applicable to the failing request. The return and reason codes are in hex. The R_GetInfo return codes are documented in *z/OS Security Server RACF Callable Services*.

Programmer response: Use the return codes to resolve the R_GetInfo problem and try the service again.

System action: The called function fails.

ITY6027 Error retrieving EIM configuration information from the caller's USER profile. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_USER_XTR (6027)

Explanation: The EIM API failed while retrieving EIM information from RACF. An R_GetInfo callable service error occurred while retrieving the EIM segment from the caller's USER profile. Failing user ID and return codes appear in the EimRC substitution text. The substitution text is:

```
USER(user ID) SAF RC(xxxxxxxx) RACF RC(xxxxxxxx)
RACF RSN(xxxxxxxx)
```

The return and reason codes are in hex. The R_GetInfo return codes are documented in *z/OS Security Server RACF Callable Services*.

Programmer response: Use the return codes to resolve the problem in the user EIM segment and try the service again. If the substitution text does not contain values for SAF RC, RACF RC, and RACF RSN, the required profile or segment was not found.

System action: The called function fails.

ITY6028 Error retrieving EIM information from a RACF profile. %1$s

Symbolic Identifier (value): EIMERR_ZOS_XTR_EIM (6028)

Explanation: The EIM API failed while retrieving EIM information from RACF. An R_GetInfo callable service error occurred while retrieving the EIM segment from a RACF profile. Failing user ID, class, profile name and return codes appear in the EimRC substitution text. The substitution text is:

```
USER(user ID) CLASS(class) PROFILE(profile name)
SAF RC(xxxxxxxx) RACF RC(xxxxxxxx) RACF
RSN(xxxxxxxx)
```

The return and reason codes are in hex. The R_GetInfo return codes are documented in *z/OS Security Server RACF Callable Services*.

Programmer response: Use the return codes to resolve the problem in the profile and try the service again. If the substitution text does not contain values for SAF RC, RACF RC, and RACF RSN, the required profile or segment was not found.

System action: The called function fails.

ITY6029 Error retrieving PROXY information from a RACF profile. %1$s

Symbolic Identifier (value):
EIMERR_ZOS_XTR_PROXY (6029)

Explanation: The EIM API failed while retrieving PROXY information from RACF. An R_GetInfo callable service error occurred while retrieving the PROXY segment from a RACF profile. Failing user ID, class, profile name and return codes appear in the EimRC substitution text. The substitution text is:

```
USER(user ID) CLASS(class) PROFILE(profile name)
SAF RC(xxxxxxxx) RACF RC(xxxxxxxx) RACF
RSN(xxxxxxxx)
```

The return and reason codes are in hex. The R_GetInfo return codes are documented in *z/OS Security Server RACF Callable Services*.

Programmer response: Use the return codes to resolve the problem in the profile and try the service again. If the substitution text does not contain values for SAF RC, RACF RC, and RACF RSN, the required profile or segment was not found.

System action: The called function fails.

ITY6500E com.ibm.ictx.identitycontext.zOS StorageMechanism is only valid on z/OS release 1.8 or later systems.

Explanation: The zOSStorageMechanism was instantiated on an unsupported system.

User response: Utilize the StorageMechanismFactory and the LdapStorageMechanism for remote access to the identity cache.

System action: Processing stops.

ITY6501E Length of ″{0}″ is not valid. Length found: {1}. Length must be exactly: {2}.

Explanation: A specified parameter has an invalid length.

User response: Correct the invalid length and retry the method.

System action: Processing stops.

ITY6502E The R_cacheserv callable service failed. {0}

Explanation: The method failed while attempting to call the R_cacheserv callable service. The returned exception text will contain the return and reason codes from R_cacheserv. The substitution text is:

```
SAF RC(return code) RACF RC(return code)
RACF RSN(reason code)
```

User response: Use the return and reason codes to

resolve the R_cacheserv problem and try the method again.

System action: Processing stops.

ITY6503E The R_cacheserv callable service failed. Not authorized to use this service. {0}

Explanation: The method failed while attempting to call the R_cacheserv callable service. R_cacheserv return and reason codes indicate an authorization failure. The returned exception text will contain the return and reason codes from R_cacheserv. The substitution text is:

```
SAF RC(return code) RACF RC(return code)
RACF RSN(reason code)
```

User response: Use the return and reason codes to resolve the R_cacheserv problem and try the method again.

System action: Processing stops.

ITY6504E The R_cacheserv callable service failed. The length of the identity context record is not valid. {0}

Explanation: The method failed while attempting to call the R_cacheserv callable service. R_cacheserv return and reason codes indicate that the identity context is too large to store successfully. The returned exception text will contain the return and reason codes from R_cacheserv. The substitution text is:

```
SAF RC(return code) RACF RC(return code)
RACF RSN(reason code)
```

User response: Use the return and reason codes to resolve the R_cacheserv problem and try the method again.

System action: Processing stops.

ITY6505E Error calling EIM API: ″{0}″. EIM error message: ″{1}″

Explanation: The method failed while attempting to call an EIM API. The returned exception will include substitutions for the failing EIM API name and the EIM error message.

User response: Use the EIM API name and error message to resolve the EIM problem and try the method again.

System action: Processing stops.

ITY6506E Error calling EIM API: ″{0}″. EIM Error Message could not be accessed. eimErr2String errno text: ″{1}″

Explanation: The method failed while attempting to call an EIM API. Another error occurred while calling eimErr2String to access the EIM message. The returned exception will include substitutions for the failing EIM API name and the eimErr2String errno text.

User response: Use the EIM API name and eimErr2String errno text to resolve the EIM problem and try the method again.

System action: Processing stops.

ITY6507E Configuration prevents store requests: USEMAP(NO) DOMAP(NO) MAPREQUIRED(YES).

Explanation: The current ICTX configuration does not allow store requests. MAPREQUIRED(YES) requires a local mapping, but both USEMAP and DOMAP are disabled.

User response: If a local mapping is desired, enable USEMAP and/or DOMAP. If a local mapping is not desired, disable MAPREQUIRED.

System action: Processing stops.

ITY6508E Configuration allows only Type 1 authentication contexts be stored: MAPREQUIRED(YES).

Explanation: The current ICTX configuration requires a local mapping but the provided identity context was not a Type 1 authentication context. In order to perform a local mapping, information must be parsed from a Type 1 authentication context.

User response: If a local mapping is desired, a Type 1 authentication context must be provided. If an alternate identity context is to be stored, MAPREQUIRED must be disabled.

System action: Processing stops.

ITY6509E Configuration prevents store requests: Local registry must be configured with MAPREQUIRED(YES).

Explanation: The current ICTX configuration indicates local mapping is required but the local registry is not configured. In order to perform a local mapping with USEMAP or DOMAP, the local registry must be configured.

User response: If a local mapping is desired, use RACF commands to configure the local registry. If a local mapping is not desired, disable MAPREQUIRED.

System action: Processing stops.

ITY6510E Store request failed. Configuration requires local mapping: MAPREQUIRED(YES). {0} {1}

Explanation: The current ICTX configuration indicates local mapping is required but both USEMAP and DOMAP could not be successfully completed. The

substitution text indicates the reasons why USEMAP and DOMAP could not be completed.

USEMAP substitutions:

- Configuration does not allow premapped data: USEMAP(NO).
- Premapped data was not provided.
- The registry from the premapped data did not match the configured local registry.
- The user from the premapped data does did have a valid length. Valid values are 1 to 8.

DOMAP substitutions:

- Configuration does not allow local mapping: DOMAP(NO).
- Local mapping was not found.
- Multiple local mappings were found.
- Local mapping found, but length is not valid. Valid values are 1 to 8.
- Could not perform local mapping. Source user was not found.
- Could not perform local mapping. Source registry was not found.

User response: If a local mapping is desired, use the exception text to resolve the USEMAP and/or DOMAP problem and try the method again. If a local mapping is not desired, disable MAPREQUIRED.

System action: Processing stops.

ITY6511E Error processing returned identity context. RecordName size is not valid.

Explanation: A length inconsistency was detected in an internal identity context data structure.

User response: If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6512E Error processing returned identity context. RecordName eyecatcher not valid.

Explanation: A invalid data structure identifier (eyecatcher) was detected in an internal identity context data structure.

User response: If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6513E Error processing returned identity context. RecordName version not valid.

Explanation: An invalid version was detected in an internal identity context data structure.

User response: If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6514E Error processing returned identity context. No Context or record name returned.

Explanation: An error was detected while attempting to retrieve an identity context.

User response: If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6515E Incompatible object class detected for parameter: {0}

Explanation: A parameter is not of the expected class type. The returned exception will include a substitution for the incompatible parameter.

User response: Ensure all parameters match the method signature. Use parameter name to resolve the problem and try the method again.

System action: Processing stops.

ITY6516E An exception was caught while calling an LdapContext method: {0}. Root exception: {1}

Explanation: An exception was caught while calling an LdapContext method. The returned exception will include substitutions for the failing LdapContext method name and the root exception text.

User response: Use the LdapContext method name and root exception text to resolve the problem and try the method again.

System action: Processing stops.

ITY6517E Error parsing data returned from the ICTX LDAP server. ExopData size is not valid.

Explanation: A length inconsistency was detected in an internal identity context data structure.

User response: If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6518E Error parsing data returned from the ICTX LDAP server. ExopData version not valid. Version found: {0}

Explanation: An invalid version was detected in an internal identity context data structure.

User response: If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6519E Error parsing data returned from the ICTX LDAP server. ExopData eyecatcher not valid. Version found: {0}

Explanation: An invalid data structure identifier (eyecatcher) was detected in an internal identity context data structure.

User response: If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6520E Error parsing returned data. IctxRc version not valid. Version found: {0}

Explanation: An invalid version was detected in an internal identity context data structure.

User response: If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6521E An exception was caught while attempting codepage convert to encoding: {0}. Root exception: {1}

Explanation: An exception was caught while attempting a codepage conversion. The returned exception will include substitutions for the encoding name and root exception text.

User response: Use the encoding name and root exception text to resolve the problem and try the method again. Ensure the international version of the JRE, which contains the Extended Encoding Set (contained in lib/charsets.jar), has been installed.

System action: Processing stops.

ITY6522E Unable to load JNI library: {0}.

Explanation: An error was detected while attempting to load a JNI library. The returned exception will include a substitution for the JNI library name.

User response: JNI libraries are operating system specific. Ensure that the caller is on a supported OS and release. Ensure that the JNI library is in the caller's PATH and that the caller has at least read access. Retry the method.

System action: Processing stops.

ITY6523E Error detected while retrieving configuration information. RCVI pointer is not valid.

Explanation: An error was detected while attempting to load configuration information. An invalid pointer to the RCVI configuration information was detected.

User response: Attempt to refresh the configuration information by RACLIST REFRESHing the LDAPBIND class and try the method again. If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6524E Error detected while retrieving configuration information. Active table pointer is not valid.

Explanation: An error was detected while attempting to load configuration information. An invalid pointer in the RCVI was detected.

User response: Attempt to refresh the configuration information by RACLIST REFRESHing the LDAPBIND class and try the method again. If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6525E Error detected while retrieving configuration information. Version is not valid.

Explanation: An error was detected while attempting to load configuration information. An invalid version in the RCVI was detected.

User response: Attempt to refresh the configuration information by RACLIST REFRESHing the LDAPBIND class and try the method again. If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6526E Error processing R_dcekey results. BINDPW returned with zero length.

Explanation: The method failed while attempting to parse BINDPW configuration information returned from the R_ dcekey callable service.

User response: Ensure an appropriate ICTX profile is defined and the LDAPBIND class is active and RACLISTed. If the profile is updated, the LDAPBIND class must be RACLIST REFRESHed for the changes to become active.

System action: Processing stops.

ITY6527E The R_dcekey callable service failed. {0}

Explanation: The method failed while attempting to call the R_dcekey callable service. The returned exception text will contain the return and reason codes from R_dcekey. The substitution text is:

```
SAF RC(return code) RACF RC(return code)
RACF RSN(reason code)
```

User response: Use the return and reason codes to resolve the R_dcekey problem and try the method again.

System action: Processing stops.

ITY6528E The R_dcekey callable service failed. Bind password is missing. {0}

Explanation: The method failed while attempting to call the R_ dcekey callable service. R_ dcekey return and reason codes indicate that the bind password is not configured. The returned exception text will contain the return and reason codes from R_ dcekey. The substitution text is:

```
SAF RC(return code) RACF RC(return code)
RACF RSN(reason code)
```

User response: Use the return and reason codes to resolve the R_ dcekey problem and try the method again. Ensure an appropriate ICTX profile is defined and the LDAPBIND class is active and RACLISTed.

System action: Processing stops.

ITY6529E Required LDAP bind information not configured: {0}

Explanation: An error was detected while attempting to load default bind credentials. A field required to bind to LDAP was not configured. The returned exception text will contain the name of the required field.

User response: Ensure an appropriate ICTX profile is defined and the LDAPBIND class is active and RACLISTed. If the profile is updated, the LDAPBIND class must be RACLIST REFRESHed for the changes to become active.

System action: Processing stops.

ITY6530E Error processing returned identity context. OidContext name size is not valid.

Explanation: A length inconsistency was detected in an internal identity context data structure.

User response: If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6531E Error processing returned identity context. OidContext eyecatcher not valid.

Explanation: An invalid data structure identifier (eyecatcher) was detected in an internal identity context data structure.

User response: If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6532E Error processing returned identity context. OidContext version not valid.

Explanation: An invalid version was detected in an internal identity context data structure.

User response: If the problem recurs, contact the IBM support center.

System action: Processing stops.

ITY6592A Suffix *suffix_name* already in use. Remove duplicate suffix definition.

Explanation: The LDAP server reported that *suffix_name* was already defined as a suffix for another plugin or backend when ICTX plugin initialization attempted to register that name. The ICTX plugin must be defined with suffix “CN=ICTX”, which should not be used by any other plugin or backend in the configuration file. Neither should any subordinate suffix value in any letter case such as “cn=abc, cn=ictx” be defined. Also verify the ICTX configuration statement is not duplicated accidentally.

User response: Correct the configuration file statements and restart the LDAP server.

System action: ICTX plugin initialization fails. The LDAP server stops.

ITY6593A Unexpected function error encountered during ICTX initialization.

Explanation: The ICTX plugin initialization process was terminated prematurely as a result of an unforeseen error. Look for associated diagnostic information in the system or job log. Consider restarting the LDAP server with tracing enabled to obtain additional error details.

User response: Correct cause of the error and restart the LDAP server.

System action: ICTX plugin initialization fails. The LDAP server stops.

ITY6594A **Incorrect plugin type configured for ICTX. Specify type clientOperation instead.**

Explanation: The ICTX plugin was defined in the configuration file with plugin type other than clientOperation.

User response: Ensure clientOperation is specified and restart the LDAP server.

System action: ICTX plugin initialization fails. The LDAP server stops.

ITY6595E **Failure encountered in the audit logging facility. R_auditx returned** *saf_return_code*, *racf_return_code*, *racf_reason_code*.

Explanation: ICTX extended operations encountered a failure when invoking the audit logging facility. The R_auditx callable service returned the values *saf_return_code*, *racf_return_code*, and *racf_reason_code*.

User response: If the remote auditing function is needed, correct the problem indicated by the R_auditx codes. R_auditx code set ″8, 8, 4″ indicates the user associated with the LDAP server does not have at least READ access to the FACILITY class profile IRR.RAUDITX.

System action: Initialization continues, but the remote auditing requests will fail until an administrator responds to the problem.

ITY6596A **Incorrect suffix configured for ICTX. Specify suffix ″CN=ICTX″ instead.**

Explanation: ICTX extended operations encountered an error verifying configuration information. A suffix other than ″CN=ICTX″ was specified

User response: Ensure only the one ″CN=ICTX″ suffix is specified and restart the IBM TDS server.

System action: ICTX extended operations initialization fails. The IBM TDS server stops.

ITY6597A **Too many suffixes configured for ICTX. Specify only suffix ″CN=ICTX″.**

Explanation: ICTX extended operations encountered an error verifying configuration information. More than one suffix was specified.

User response: Ensure only the one ″CN=ICTX″ suffix is specified and restart the IBM TDS server.

System action: ICTX extended operations initialization fails. The IBM TDS server stops.

ITY6598A **No suffix configured for ICTX. Specify suffix ″CN=ICTX″.**

Explanation: ICTX extended operations encountered an error verifying configuration information. The required suffix ″CN=ICTX″ was not specified.

User response: Specify suffix ″CN=ICTX″ and restart the IBM TDS server.

System action: ICTX extended operations initialization fails. The IBM TDS server stops.

ITY6598A **ICTX suffix must be specified exactly once.**

Explanation: ICTX extended operations encountered an error verifying configuration information. The required suffix ″CN=ICTX″ was not specified, or additional suffixes were specified.

User response: Correct the ICTX suffix specification and restart the LDAP server.

System action: ICTX extended operations initialization fails. The LDAP server stops.

ITY6599E **Failure to update ICTX cache. R_cacheserv returned** *saf_return_code*, *racf_return_code*, *racf_reason_code*.

Explanation: ICTX extended operations encountered a failure when attempting to store an entry in the cache. The R_cacheserv callable service returned the values *saf_return_code*, *racf_return_code*, and *racf_reason_code*.

User response: If ICTX cache functions are needed, correct the problem indicated by the R_cacheserv codes. R_cacheserv code set ″8, 8, 16″ indicates the user associated with the LDAP server does not have UPDATE access to the FACILITY class profile IRR.RCACHESERV.ICTX.

System action: ICTX extended operations initialization fails. The LDAP server stops.

Chapter 9. The eimadmin utility

The eimadmin utility is a z/OS UNIX System Services Shell tool. An administrator can use it to define an EIM domain and prime the domain with registries, identifiers, and associations between identifiers, registry users and policies. An administrator can also use eimadmin to give users (and other administrators) access to an EIM domain or list or remove the EIM entities.

An administrator can use the **eimadmin** command in two ways:

- By including information with command-line options on an **eimadmin** command
- By including information in an input file that an **eimadmin** command references

You can create the file manually or by exporting records from a data base. (See “Using an input file” on page 128 for more information.) The administrator directs utility processing by specifying a combination of command-line options. (See “Purpose” on page 110 and “Parameters” on page 114 for more information.)

eimadmin

Purpose

Perform actions on the following objects:

- Domains
- Registries
- Identifiers
- Associations
- Access authorities
- Policies

The actions you can perform include the following:

- Add an object
- Purge an object
- List objects (for example, list directories, list registries, and so forth)
- Modify attributes associated with objects
- Erase attributes

Format

```
eimadmin -a | -p | -l | -m | -e
         -D | -R | -I | -A | -C  | -Y
                  [-b bindDN]
      [-B attribute]
      [-c accessType]
              [-d domainDN]
      [-E certificate]
      [-f accessUserType]
              [-F issuerFilter]
      [-g registryParent]
              [-G certificateFilterTemplate]
      [-h ldapHost]
      [-i identifier]
              [-j otherIdentifier]
      [-J subjectFilter]
      [-k URI]
              [-K keyFile [ -P keyFilePassword] [-N certificateLabel]]
      [-n description]
              [-o information]
              [-q accessUser]
              [-r registryName]
                [-s switch]
                [-S connectType]
      [-t associationType]
              [-T targetRegistry]
      [-U identifierUUID]
      [-u registryUser]
              [-v verboseLevel]
      [-w bindPassword]
      [-x registryAlias]
              [-y registryType]
              [-z registryAliasType]
```

Table 29 on page 111 summarizes the objects and actions and the flags required and optional for each.

Tips:

- Each **eimadmin** command must include one action and one object type. Depending on the object and action you are performing on it, EIM might require additional parameters.
- Some options are for multi-value attributes, which you can specify more than once. Other options are for single-value attributes, which you can specify **once**. (If you repeat an option that is for a single-value attribute, eimadmin processes only the first value it encounters in the command.) Other than this, the order in which you specify parameters is not important.
- You can code the parameters of the **eimadmin** command in several ways:
  - You can concatenate an action and an object, but must omit the embedded hyphen:

    ```
    -aD
    ```

  - You can include both hyphens but must separate the two options with a space:

    ```
    -a -D
    ```

  - In other words, the following is **not** valid because it includes both hyphens and there is no space before `-D`:

    ```
    -a-D
    ```

The following table summarizes required and optional flags for each object type and action pair. You can specify the value for most options in an input file instead of specifying it on the command line. See “Using an input file” on page 128 for more information. See Table 33 on page 130 for the mapping of file labels with command line options. **Rule:** The required connection flags, generally independent of the specified object type and action, are shown in Table 29.

Table 29. Required and optional flags

Object	Action	Required	Optional	Comments
D	a	b,d, h, w	n, B	Add a domain. **b** and **w** can be omitted if bind information and password are obtained from the RACF profile.
	p	d, h, b, w	s	Remove a domain. If the domain is not empty, include **'-s RMDEPS'**.
	l	d, h, b, w		List domain(s). Specify **-d'*'** to list all domains.
	m	d, h, b, w	n, B	Modify or add a domain attribute.
	e	d, h, b, w	n, B	Remove or clear a domain attribute.
R	a	r,y	g, k, n, x, z, B	Add a registry. The value specified for **'-r'** is assumed to be a new system registry unless **'-g'** is also specified, in which case the **'-r'** value indicates a new application registry.
	p	r	s	Remove a registry.
	l	r	y	List registries. Return all registry entries in the domain that match the specified **'-r'** value search filter, which might contain the wild card **'*'**.
	m	r	k, n, x, z, B	Modify or add a registry attribute, including a registry alias.
	e	r	k, n, x, z, B	Remove or clear a registry attribute, including a registry alias.

Table 29. Required and optional flags (continued)

Object	Action	Required	Optional	Comments
I	a	i	j, n, o	Add an identifier.
	p	i		Remove an identifier.
	l	i		List an identifier by unique identifier name. Return all identifier entries in the domain that match the specified **'-i'** value search filter, which might contain the wild card **'*'**.
		U		Echo the identifier UUID followed by the unique identifier name. The **-U** value search filter cannot include the wild card **'*'**.
		j		List an identifier by non-unique identifier name. Return all identifier entries in the domain that have a non-unique identifier matching the specified **'-j'** value search filter, which might contain the wild card **'*'**.
	m	i	j, n, o	Modify or add an identifier attribute.
	e	i	j, n, o	Remove or clear an identifier attribute.
A	a	i, r, u or E, t	n, o	Add an association. You can repeat the **'-t'** option to add multiple associations types. Flags **'-n'** and **'-o'** are relevant only to TARGET associations.
	p	i, r, u or E, t		Remove an association. You can repeat the **'-t'** option to remove multiple associations types.
	l	i	t	List association(s). Return all associations in the domain for specified **'-i'** unique identifier. Specify a **'-t'** value to limit the entries returned to the given association type.
	m	r, u or E, t	n, o	Modify or add an association attribute. Flags **'-n'** and **'-o'** are relevant only to TARGET associations.
	e	r, u or E, t	n, o	Remove or clear an association attribute. Flags **'-n'** and **'-o'** are relevant only to TARGET associations.

Table 29. Required and optional flags (continued)

Object	Action	Required	Optional	Comments
Y	a	t *DOMAIN*, T, u or E	n, o	Add a default domain policy association. For association type **-t** DOMAIN only.
		t *REGISTRY*, r, T, u or E	n, o	Add a default domain policy association. For association type **-t** REGISTRY only.
		r, one or more of J,F, G		Add a filter policy. No association type is specified.
		t *FILTER*, r one or more of J, F, G, T, u or E	n, o,	Add a filter policy and a filter policy association. For association type **-t**, FILTER only.
	p	t *DOMAIN*, T, u or E		Remove a default domain policy association. For association type **-t** DOMAIN only.
		t *REGISTRY*, r,T, u or E		Remove a default registry policy association. For association type **-t** REGISTRY only.
		r, one or more of J, F, G		Remove a filter policy and it's filter policy associations. No association type is specified.
		-t FILTER, r, one or more of J, F, G, T, u or E		Remove a filter policy association. For association type **-t** FILTER only.
	l	t *POLICY*	r, T	List all types of policies for a domain, all policies with the source registry, and/or all policies with the target registry. For association type **-t** POLICY.
		t *DOMAIN*	T, u or E	List default registry policies. For association type **-t** DOMAIN only.
		t *REGISTRY*	r, T, u or E	List default registry policies. For association type **-t** REGISTRY only.
		t *FILTER*	T, u or E, J, F, G, r	List only filter policies with associations. For association type **-t** FILTER only.
			R, J, F, G	List all filters policies. Associations are not displayed. No association type is specified.
	m	T, u or E	n, o	Modify or add an attribute belonging to the target user of a policy.
	e	T, u or E	n, o	Remove or clear an attribute belonging to the target user of a policy.
C	a	c, q, f	r	Add access. For access type REGISTRY, provide a specific **'-r'** registry value, or a wild card **'*'** indicating access to all registries in the domain.
	p	c, q, f	r	Remove access. For access type REGISTRY, provide a specific **'-r'** registry value, or a wild card **'*'** indicating access to all registries in the domain.
	l	c	r	List access by type. For access type REGISTRY, provide a specific **'-r'** registry value, or a wild card **'*'** indicating access to all registries in the domain.
		q, f		List access by user.

Parameters

Actions

-a|-p|-l|-m|-e
This is the action you want to perform:

-a Add an object. (Create an object definition and its attributes.)

-p Purge an object. (Remove an object definition and its attributes.)

-l List an object. (Retrieve an object definition and its attributes.)

-m Modify an attribute. (Alter an attribute of an existing object, either by changing a single-value attribute or adding a multi-value attribute.)

-e Erase an attribute. (Clear a single-value attribute or remove a multi-value attribute.)

Object types

-D|-R|-I|-A|-C |-Y
This parameter specifies the object types on which to perform the action:

-D A domain. This is a collection of identifiers, user registries, and associations between identifiers and user IDs, and policies, stored within an LDAP directory. (For more information about EIM domains, see page “EIM domain” on page 8.)

-R A registry. This is the name of a user registry. Associations are defined between identifiers and user IDs in the user registry. It is a logical collection of user identities and policies. (For more information, see page “EIM domain” on page 8.)

-I An identifier. This is the name of a person or entity participating in an EIM domain. (For more information, see page “EIM domain” on page 8.)

-A An association. This is a relationship between an identifier in the EIM domain with a user ID. (For more information, see page “EIM domain” on page 8.)

-C An access authority. This is an EIM-defined LDAP access control group. (For more information, see page “EIM access control” on page 30.)

-Y A policy. This is a relationship between a registry or domain and a user identity in a target registry.

Processing controls, attributes, and connection values

Processing controls

Processing controls include the following:

-s *switch*
The *switch* specifies a value that affects the way the eimadmin utility functions operate. You can specify the following value:

RMDEPS
Remove dependents when removing a domain or system registry. This facilitates removing a domain by first removing all identifiers and registries defined for

the domain. It facilitates the removing a system registry by first removing all applications registries defined for the registry.

Attention: The eimadmin utility does not warn you that dependents exist before removing them, so use this switch carefully.

-U *identifierUUID*
The *identifierUUID* is the universally unique identifier form of the EIM identifier. This flag is only valid with `eimadmin -lI`.

-v *verboseLevel*
The *verboseLevel* is an integer from 1 to 10, that controls the amount of trace detail that the eimadmin utility displays. (It is for diagnosing problems in the eimadmin utility.) The default value of 0 indicates no trace information. You can specify an integer value from 1 to 10, from the least to greatest amount of trace information.

The utility checks the value and displays trace information defined for the level and all lower levels. The following levels trigger specific information:

- ″3″, which indicates EIM API call parameters and return values
- ″6″, which indicates option values and input file labels
- ″9″, which indicates utility routine entry and exit statements

Objects and attributes

Rule: Options are single-valued unless indicated otherwise.

The section that follows explains required and optional attributes and their parameters.

Tips:

- You can specify these attributes as command options or as fields in input files. If you are specifying command options, you must enclose values with imbedded blanks within quotation marks (″) or (’). Quotation marks are optional for single-word values. Specifying a multi-word value without quotation marks in effect truncates the command line options; values after the first word are truncated.
- The following special characters are not allowed in *registryName*, *registryParent*, or *identifier*:

  `, = + < > # ; \ *`

Rule: Except where indicated, the parameters are single-value options. If you specify an option more than once, the utility processes only the first occurrence.

-B *attribute*
The attribute flag is used to add, modify, remove, or clear and attribute for the domain. It may also enable or disable a function in the EIM domain.

The attribute flag specified with the domain object may have the following value:

POLICY
Enable or disable default domain and registry policies. By default, default policies are disabled in the domain.

The attribute flag specified with the registy object may have one of the following values:

LOOKUP
Enable or disable lookup operations

POLICY
Enable or disable default registry policies

By default, registries are enabled for lookup operations, but default registry policies are disabled. Specifying **-e** deactivates the function. Specifying **-m** activates the function.

-c *accessType*
The *accessType* specifies the scope of access authority that a user has over the EIM domain. It must be one of the following values:

ADMIN
Specifies administrative access.

REGISTRY
Specifies registry access. If you specify REGISTRY, you must also specify a registry value (-r). The registry value can be a specific registry name or it can be an asterisk (*) to indicate access to all registries.

IDENTIFIER
Specifies identifier access.

MAPPING
Specifies mapping operations access.

-E *certificate*
The name of a file or dataset containing an X.509 certificate. The name stored in the certificate will be used to create an association with an identifier. The certificate may be DER or base 64 encoded. The base 64 encoded certificate may be in EBCIDIC or ASCII.

When working with an source, target, or administrative associations, the subject's distinguished name, the issuer's distinguished name, and the public key info are extracted from the certificate and used to create the registry user name for the association.

-f *accessUserType*
The *accessUserType* specifies the type for the access user name. It must be one of the following:

DN
The *accessUser* is a distinguished name. (See page 118 for a description of *accessUser*.)

KERBEROS
The *accessUser* is a Kerberos identity. (See page 118 for a description of accessUser.)

-F *issuerFilter*
All or part of the issuer portion of a certificate filter policy name. When

used without the **-E** *certificate* flag, the issuer filter must be a sequence of relative distinguished names that make up a complete certificate filter policy issuer name. For example:

```
O=A Certificate Authority,L=Internet
```

When used with the **-E** *certficate* flag, the issuer filter is a substring that indicates what part of the issuer's distinguished name from the certificate should be used for the certificate filter policy's issuer name. For example if the certificate has the issuer's distinguished name of the following:

```
OU=Certified User,O=A Certificate Authority,L=Internet
```

and the issuerFilter value is O=, then the issuer portion of the certificate filter policy name is:

```
O=A Certificate Authority,L=Internet
```

If the **-F** flag is omitted when working with certificate filter policies, the certificate filter policy's name will not contain an issuer filter.

-G *certificateFilterTemplate*
The name of a file or dataset containing an X.509 certificate. The name stored in the certificate will be used to create a certificate filter policy. The certificate may be DER or base 64 encoded. The base 64 encoded certificate may be in EBCIDIC or ASCII.

When working with a certificate filter policy, only the subject's distinguished name and issuer's distinguished name are extracted from the certificate. They are used as a template for the name of the certificate filter policy. The **-J** *subjectFilter* and **-F** *issuerFilter* flags are used to indicate what portion of the distinguished names to use.

-g *registryParent*
The *registryParent* specifies the name of a system registry. An application registry is a subset of a system registry. If you are adding an application registry, you must use the **-r** option and the **-g** option. The **-r** value is the application registry you are defining. The **-g** option is the preexisting system registry.

-i *identifier*
The *identifier* is a unique identifier name.

Example:

```
John Day
```

-j *otherIdentifier*
The *otherIdentifier* specifies a non-unique identifier name.

Example:

```
John
```

Note: You can specify this option multiple times to assign multiple non-unique identifiers.

-J subjectFilter
All or part of the subject portion of a certificate filter policy name. When used without the **-E** *certificate* flag, the subject filter must be a sequence of relative distinguished names that make up a complete certificate filter policy subject name. For example:

```
O=My Company,C=US
```

When used with the **-E** *certificate* flag, the subject filter is a substring that indicates what part of the subject's distinguished name from the certificate should be used for the certificate filter policy's subject name. For example if the certificate has the following subject distinguished name:

```
CN=John Smith,O=My Company,C=US
```

and the subjectFilter value is C=, then the subject portion of the certificate filter policy name is:

```
C=US
```

If the **-J** flag is omitted when working with certificate filter policies, the certificate filter policy name will not contain a subject filter.

-k *URI*
The *URI* specifies the Universal Resource Identifier (URI) for the registry (if one exists).

-n *description*
The *description* specifies any text (that you provide) to associate with the domain, registry, identifier, or association.

Note: You can define a user description only for target associations.

-o *information*
The *information* specifies additional information to associate with an identifier or association.

Note: You can define user information only for target associations. You can specify this option multiple times to assign multiple pieces of information.

-q *accessUser*
The *accessUser* specifies the user distinguished name (DN) or the Kerberos identity with EIM access, depending on the *accessUserType* specified.

-r *registryName*
The *registryName* specifies the name of a registry. When you add a new registry, eimadmin considers the registry a system registry unless you also specify the **-g** option. If you specify the **-g** option, eimadmin considers the registry an application registry.

-t *associationType*
The *associationType* specifies the relationship between an identifier and a registry or a policy type. It must be one of the following:

ADMIN
Indicates associating a user ID with an identifier for administrative purposes.

SOURCE
Indicates that the user ID is the source (or from) of a lookup operation.

TARGET
Indicates that the user ID is the target (or to) of a lookup operation.

DOMAIN
Indicates a default domain policy.

REGISTRY
Indicates a default registry policy.

FILTER
Indicates a certificate filter policy.

POLICY
Indicates any kind of policy.

Note: You can specify this option multiple times to define multiple relationships.

-T *targetRegistryName*
The targetRegistryName specifies the name of a registry. This value is used when creating or deleting a default registry policy.

-u *registryUser*
The *registryUser* specifies the user ID of the user defined in the registry.

-x *registryAlias*
The *registryAlias* specifies another name for a registry.

See “Working with registry aliases” on page 61 for information about working with aliases. You can specify this option multiple times to assign multiple aliases.

-y *registryType*
The *registryType* specifies the type of registry. Predefined types that eimadmin recognizes include the following:

- RACF
- OS400
- KERBEROS (for case ignore)
- KERBEROSX (for case exact)
- AIX
- NDS
- LDAP
- PD (Policy Director)
- WIN2K
- X509
- LINUX
- DOMINOS
- DOMINOL

You can also create your own types by concatenating a unique OID with one of the following two normalization methods:

- -caseIgnore
- -caseExact

(See “EIM registry definition” on page 13 for more information.)

-z *registryAliasType*
The *registryAliasType* specifies the type for a registry alias. You can invent your own value or use one of the following suggested values:

- DNSHostName
- KerberosRealm

- IssuerDN
- RootDN
- TCPIPAddress
- LdapDnsHostName

Note: For a set of command line options or single input data record, the eimadmin utility recognizes only the first specification of *registryAliasType*. However, the eimadmin utility does recognize multiple registry aliases and associates all of them with the single *registryAliasType*.

Connection values

The connection information needed by the utility includes the EIM domain (-d) and its controlling server (-h), the identity (-b,-w; or -K,-P,-N) with which to authenticate (bind) to the server, and the authentication method (-S).

For object types other than domain (-D), specifying the domain, server and bind identity is optional. If not specified, the information is retrieved from a RACF profile. See “Storing LDAP binding information in a profile” on page 68 for more information.

Rule: If any of the connect information is specified, the full set of values required for the connect type must be specified. Omitting one or more values (but not all) results in an error. Table 30 shows the required and optional values for each connect and host type when specified with the eimadmin command:

Table 30. Required connection values

Connection type	Host type `secure(ldaps://)` / `non-secure(ldap://)`	Required values	Optional values
SIMPLE or CRAM-MD5	secure	-d, -h, -b, -w, -K, -P	-N
	non-secure	-d, -h, -b, -w	
EXTERNAL	secure	-d, -h, -K, -P, -S	-N
	non-secure	unsupported	unsupported
GSSAPI	secure	-d, -h, -K, -P, -S	-N
	non-secure	-d, -h, -S	

Tips:

- Exceptions:
  - The domain option (-d) is not required for domain functions if the value is specified through an input file.
  - An SSL key database file password or stash file (-P) is not required when -K specifies a RACF key ring.
- The utility prompts for the simple bind password if required and -w is not specified on the command line, and prompts for the SSL key database file password if required and -P is not specified on the command line.

-S *connectType*

The *connectType* is the method of authentication to the LDAP server. It must be one of the following values:

- SIMPLE (bind DN and password)
- CRAM-MD5 (bind DN and protected password)
- EXTERNAL (digital certificate)

- GSSAPI (Kerberos)

If not specified, the connect type defaults to SIMPLE.

For connect type GSSAPI, the default Kerberos credential is used. This credential must be established using a service such as **kinit** prior to running eimadmin. For **kinit** and related information, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

-d *domainDN*

The *domainDN* is the full distinguished name (DN) of the EIM domain. It begins with 'ibm-eimDomainName='. It further consists of:

- *domainName* — The name of the EIM domain you are creating, for example: `MyDomain`
- *parent distinguished name* — The distinguished name for the entry immediately above the given entry in the directory information tree hierarchy, for example, ″o=ibm,c=us″.

Example:

```
ibm-eimDomainName=MyDomain,o=ibm,c=us
```

-h *ldapHost*

The *ldapHost* is the URL and port for the LDAP server controlling the EIM data. The format is:

Example:

```
ldap://some.ldap.host:389
ldaps://secure.ldap.host:636
```

-b *bindDN*

The *bindDN* is the distinguished name to use for the simple bind to LDAP. The format is:

Examples:

```
cn=Johns Admin
```

or

```
cn=Johns Admin,o=ibm,c=us
```

-w *bindPassword*

The *bindPassword* is the password associated with the bind DN (for the LDAP bind).

-K *keyFile*

The *keyFile* is the name of the SSL key database file, including the full path name. If the file cannot be found, it is assumed to be the name of a RACF key ring which contains authentication certificates. This value is required for SSL communications with a secure LDAP host (prefixed `ldaps://`).

Example:

```
/u/eimuser/ldap.kdb
```

-P *keyFilePassword*

The keyFilePassword is the password required to access the encrypted information in the key database file. Alternatively, you can specify an SSL password stash file for this option by prefixing the stash file name with `file://`.

Example:

```
secret
or
file:///u/eimuser/ldapclient.sth
```

Note: The eimadmin utility prompts for a key file password if you specify the name of a key database file for -K but not the -P option on the command line.

-N *certificateLabel*
The *certificateLabel* identifies which certificate to use from the key database file or RACF key ring. If this option is not specified, the certificate marked as the default in the file or ring is used.

Example:

```
eimcert
```

Authorization

The LDAP administrator has the authority to use the eimadmin utility and access to all the functions it provides. EIM administrators can use the utility as long as:

- They have a bind distinguished name and password defined at the LDAP server containing the EIM domain
- Their bind distinguished name has one of the EIM authorities:
  - EIM administrator
  - EIM registries administrator
  - EIM registry X administrator
  - EIM identifiers administrator

  See “EIM access control” on page 30 for details about the specific tasks each administrator can perform.

Messages

Eimadmin issues a message to prompt for a password or to indicate an error. Do not expect to receive a message for successful completion unless you use an input file. When processing records in an input file, eimadmin issues an informational message for the start, stop, and a progress message for every 50 records.

Note: Eimadmin returns one or more data lines for list (-l) requests unless it finds no matching EIM entries or the bind identity is not authorized to that data.

For eimadmin error messages, see Chapter 8, “Messages,” specifically ITY4*xxx* messages.

Error codes

The eimadmin utility returns one of the following exit codes upon completion:

Table 31. Eimadmin utility exit codes

Exit code	Meaning
0	No errors encountered.
4	One or more errors encountered but, if you specified an input file, all records were processed
8	A severe error occurred that caused processing to stop before reaching the end of an input file, if specified.

Examples for working with policies

Creating an x.509 registry

The following creates an x.509 registry:

```
eimadmin -aR -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-r myCertificateMappings -y X509
```

Enabling or disabling a registry for lookup or policy operations

A registry may be enabled for lookup operations by using the **-mR** action and **-B** LOOKUP flag. The policies in a registry may be enabled by using **-B** POLICY with the **-mR** action . For example:

```
eimadmin -mR -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-r myCertificateMappings  -B POLICY
```

A registry disabled for lookups by using **-B** LOOKUP with the **-eR** action. The policies in a registry may be disabled by using **-B** POLICY with the **-eR** action. By default lookups are enabled and policies are disabled.

Enabling or disabling a domain's use of policies

```
eimadmin -mD -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-B POLICY
```

Policies may be disabled for a domain by using the **-eD** action and **-B** POLICY flag. By default, policies in a domain are disabled. Domains are always enabled for lookup operations.

Creating an association using the name stored within a certificate

This examples assumes the certificate contains the following subject's and issuer's distinguished names:

```
SDN: CN=John Day,O=My Company,C=US
IDN:  OU=Certified User,O=A Certificate Authority,L=Internet

eimadmin -aA -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-i "John Day"
-r myCertificateMappings -E certificate
-t admin
```

Listing an association that was created using a certificate

```
eimadmin -lA -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-i "John Day" -t admin
```

This produces output such as the following:

```
unique identifier: John Day
      association: ADMIN
         registry: myCertificateMappings
    registry type: X509
```

```
    registry user: <SDN>CN=John Day,O=My
Company,C=US"</SDN>OU=VeriSign Class 1 Individual Subscriber,O=A
Certificate Authority,L=Internet<
/IDN><HASH_VAL>34238b120e3675f912e3d68495847392017632ac </HASH_VAL>
```

Removing an association using the name stored within a certificate

This examples assumes the certificate contains the following subject's and issuer's distinguished names:

```
SDN: CN=John Day,O=My Company,C=US
IDN:  OU=Certified User,O=A Certificate Authority,L=Internet
```

```
eimadmin -pA -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-i "John Day"
-r myCertificateMappings -E certificate
-t admin
```

If you don't have the original certificate, another approach to removing an association involving a certificate user name is to copy the registry user name from the output of a list association and provide it with the **-u** flag. For example:

```
eimadmin -pA -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-i "John Day"
-r myCertificateMappings
-u "<SDN>CN=John Day,O=My Company,C=US</SDN>"\
"<IDN>OU=Certified User,O=ACertificate Authority,L=Internet</IDN>"\
"<HASH_VAL>34238b120e3675f912e3d68495847392017632ac</HASH_VAL>"
-t admin
```

Creating a domain policy

```
eimadmin -aY  -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-T zOSRegistry -u DAY
-t domain
```

Listing the domain policy

```
eimadmin -lY  -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-t domain
```

This produces output such as the following:

```
domain name: My Employees
             policy type: DOMAIN
          target registry: zOSRegistry
    target registry user: DAY
         domain policies: ENABLED
 target registry lookups: ENABLED
target registry policies: DISABLED
```

Deleting a domain policy

```
eimadmin -pY  -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-T zOSRegistry -u DAY
-t domain
```

Creating a registry policy

```
eimadmin -aY  -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-r zOSRegistry -T OS400Registry -u JOHNDAY
-t registry
```

Listing a registry policy

```
eimadmin -lY  -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-r zOSRegistry
-t registry
```

This produces output such as the following:

```
         source registry: zOSRegistry
             policy type: REGISTRY
         target registry: OS400Registry
    target registry user: JOHNDAY
         domain policies: ENABLED
 source registry lookups: ENABLED
 target registry lookups: ENABLED
target registry policies: DISABLED
```

Deleting a registry policy

```
eimadmin -pY  -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-r zOSRegistry -T OS400Registry -u JOHNDAY
-t registry
```

Creating a filter policy

This example assumes the file certificate contains the following subject's and issuer's distinguished names:

```
SDN: CN=John Day,O=My Company,C=US
IDN:  OU=Certified User,O=A Certificate Authority,L=Internet

eimadmin -aY  -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-r myCertificateMappings
-J "O=My Company,C=US"
-F "OU=Certified User,O=A Certificate Authority,L=Internet"
-T OS400Registry -u JOHNDAY
-t filter
```

or

```
eimadmin -aY  -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-r myCertificateMappings
-G certificate -J O= -F OU=
-T OS400Registry -u JOHNDAY
-t filter
```

A filter policy with the full subject's and issuer's distinguished names can be created using just the certificate template. For example:

```
eimadmin -aY  -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-r myCertificateMappings
-G certificate
-T OS400Registry -u JOHNDAY
-t filter
```

Listing the filter policy association

```
eimadmin -lY  -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-r myCertificateMappings
-J "O=My Company,C=US"
-F "OU=Certified User,O=A Certificate Authority,L=Internet"
-t filter
```

or

```
eimadmin -lY  -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-r myCertificateMappings
-G certificate -J O= -F OU=
-t filter
```

This produces output such as the following:

```
source registry: myCertificateMappings
             policy type: FILTER
                  filter: <SDN>CN=John Day,O=My Company,C=US</SDN><IDN>O
U=Certified User,O=A Certificate Authority,L=Internet</IDN>
         target registry: OS400Registry
    target registry user: JOHNDAY
         domain policies: ENABLED
 source registry lookups: ENABLED
 target registry lookups: ENABLED
target registry policies: DISABLED
```

Deleting a filter policy

```
eimadmin -pY  -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-r myCertificateMappings
-J "O=My Company,C=US"
-F "OU=Certified User,O=A Certificate Authority,L=Internet" -T OS400Registry
-u JOHNDAY -t filter
```

or:

```
eimadmin -pY  -h ldap://my.server
-b "cn=EIM admin,o=My Company,c=US" -w secret
-d "ibm-eimDomainName=My Employees,o=My Company,c=US"
-r myCertificateMappings -T OS400Registry -u JOHNDAY
-G certificate -J O= -F OU=
-t filter
```

Examples for listing various objects without an input file

- List a single domain by entering a command such as the following:

  ```
  eimadmin -lD -h ldap://my.server -b "cn=EIM admin,o=My Company, c=US"
  -d "ibm-eimDomainName=My Employees,o=My Company, c=US"
  ```

  This produces output such as the following:

```
domain name: My Employees
  domain DN: ibm-eimDomainName=My Employees,o=My Company, c=US
description: employees in my company
policies: DISABLED
 version: 2
```

- List a single registry by entering a command such as the following:

  ```
  eimadmin -lR -r MyRegistry
  ```

  This produces output such as the following:

  ```
            registry: MyRegistry
       registry kind: APPLICATION
     registry parent: MySystemRegistry
       registry type: RACF
         description: my racf registry
                 URI: ldap://some.big.host:389/profileType=User,cn=RACFA,o=My Company, c=US
      registry alias: TCPGROUP
  registry alias type: DNSHostName
             lookups: DISABLED
            policies: DISABLED
  ```

  Another example is:

  ```
  eimadmin -lR -h ldap://my.server
  -b "cn=EIM admin,o=My Company,c=US" -w secret
  -d "ibm-eimDomainName=My Employees,o=My Company,c=US"
  -r myCertificateMappings
  ```

  This produces output such as the following:

  ```
      registry: myCertificateMappings
         registry kind: SYSTEM
         registry type: X509
               lookups: DISABLED
              policies: DISABLED
  ```

- List identifiers by entering a command such as the following:

  ```
  eimadmin -lI -h ldap://my.server
  -b "cn=EIM admin,o=My Company, c=US" –w secret
  -d "ibm-eimDomainName=My Employees,o=My Company, c=US"
  -i "J.C.Smith"
  ```

  This produces output such as the following:

  ```
  unique identifier: J.C.Smith
    identifier UUID: d1284038asdflkae8-1395adfj379423294d
   other identifier: J.C.Smith
   other identifier: Joseph
   other identifier: Joe
        description: 004321
        information: D01
        information: 1990-04-11
  ```

- List an identifier using the identifier's UUID by entering a command such as the following:

  ```
  eimadmin -lI -h ldap://my.server
  -b "cn=EIM admin,o=My Company, c=US" –w secret
  -d "ibm-eimDomainName=My Employees,o=My Company, c=US"
  -U "d1284038asdflkae8-1395adfj379423294d"
  ```

  This produces output such as the following:

  ```
  d1284038asdflkae8-1395adfj379423294d J.C.Smith
  ```

- List target associations by entering a command such as the following:

```
eimadmin -lA -h ldap://my.server
-b "cn=EIM admin,o=My Company, c=US" –w secret
-d "ibm-eimDomainName=My Employees,o=My Company, c=US"
-i "J.C.Smith" –t target
```

This produces output such as the following:

```
    unique identifier: J.C.Smith
      identifier UUID: d1284038asdflkae8-1395adfj379423294d
              registry: MyRegistry
        registry type: RACF
          association: target
        registry user: SMITH
          description: TSO
          information: 1989-08-01
          information: ADMIN1
```

- List accesses by entering a command such as the following:

```
eimadmin -lC -c admin
```

This produces output such as the following:

```
        access user: cn=JoeUser,o=My Company, c=us

        access user: cn=admin1,o=My Company, c=us

        access user: cn=admin2,o=My Company, c=us
```

Using an input file

You can use the **eimadmin** command to add objects and associated attribute values to an EIM domain by specifying command-line options or specifying an input file name.

Tip: The advantage of using a file to add information such as associated attributes to an EIM domain is that you can input any number of entities of the same type with a single call to eimadmin. For example, you might want define a large number of identities that correspond to users in a platform-specific database such as RACF.

You can create your input file manually or export it from a database. For example, you can use IRRDBU00 to extract records from the RACF database.

Tip: To pass the input file to eimadmin, use UNIX standard input (*stdin*).

The eimadmin utility interprets the data in the input file according to the command line options you specify on an eimadmin command. For instance, if you use the `-aI` option combination, this directs eimadmin to look for identifier information within the file and add it to the EIM domain.

Input file requirements

The eimadmin requirements for the input file and its records include the following:

- The file must be sequential.
- By convention, each file line consists of a single record. (Only one record is allowed per file line. Records cannot span lines.)
- Records have a maximum length of 10,000 characters.
- Records contain column-delimited fields. (The label line defines these fields. See “The label line” on page 129 for more information.) The fields of each record contain character values representing a single object and optional associated attributes.

Tip: If you are using an input file to add identifiers, each record should contain a value that can serve as a unique EIM identifier for the user. IBM recommends that you sort the records by the field that is unique. This unique identifier might be an employee name or number, but it is unlikely to be the user ID for a registry. The identifier chosen must be unique within the EIM domain because associated user IDs cannot be unique across multiple registries.

If you have previously populated your EIM domain with unique identifiers, the unique identifiers for which you are adding associations or attributes in your input records should match these unique identifiers.

After you sort the records by the unique identifier fields, check the results to verify that non-blank values appear in this field for each record and that the values are not duplicated (unless this is intentional). The eimadmin utility generates an error each time it tries to add an object that has been previously defined.

Input file contents

The file can include:

- Comments; to include a comment line, use ″#″ as the first non-blank character in the line.

  Note: The eimadmin utility ignores blank lines and comment lines.
- Upper and lower case (The eimadmin utility preserves lettercase.)
- Blanks (Whitespace is any combination of blanks, tabs, and other 'invisible' control characters.

  Note: Avoid using whitespace characters other than blanks because the eimadmin utility does not consider them when it performs positional parsing. The eimadmin utility truncates leading and trailing whitespace within fields when it is parsing input values. The eimadmin utility does not process anything in a field that contains all whitespace.)

Table 32. Hexadecimal character values for invisible control characters

EBCIDIC Hexadecimal value	Description
05	HT — tab
0B	VT — vertical tab
0C	FF — form feed
0D	CR — carriage return

The label line

The first non-blank, non-comment line in the input file must be the label line. This consists of one or more labels that identify starting and ending positions for column-delimited fields in subsequent lines. For details about names of labels, see Table 33 on page 130.

Example:

Suppose you want to input employee records of last name, first name, and employee number.

Your data might look like the information in the following grid. (The top line of the grid is not part of the data and is there simply to show column numbers):

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
0	0	0	1		1	0	0	0		B	l	a	c	k	s	t	o	n	e		A	u	g	u	s	t	i	n	e
0	0	0	2		1	7	3	4		B	r	a	d	y							B	a	r	b	a	r	a		
0	0	0	3		1	1	2	4		L	l	o	y	d							C	a	r	o	l				
0	0	0	4		1	7	4	5		M	a	r	t	i	n	s	o	n			D	e	b	b	i	e			

(The record is 30 characters long. The first four slots are a four-digit sequence number followed by a blank. The employee numbers start in column 6 and end in column 10. The last names start in column 11 and end in column 20. The first names start in column 22 and end in column 30.)

Your label line would look like the following:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
					I	U		;		U	N								;		I	N							;

The labels in the preceding label line are:

- ″IU″ represents a unique identifier (employee number)
- ″UN″ represents the registry user name (last name)
- ″IN″ represents a non-unique identifier (first name)

These labels mark the starting positions of the fields for employee number, last name, and first name. The semicolons mark the ending positions for these fields. See “Example for adding a list of identifiers to an EIM domain” on page 132 for a more complex example.

Example — Using eimadmin with the tabular output of SMF Unload: The audit records cut by EIM contain the UUID of identifiers and policies. eimadmin can be used to get the friendly name associated with the UUID.

First, generate the tabular output using SMF Unload. The edit the tabular output so that the 'UU' label appears above the UUID. For this example, assume the data is in the HFS file smfdata.

Then run the following eimadmin command:

```
eimadmin -lI -h ldap://my.server -b "cn=EIM admin,o=My Company, c=US"
-d "ibm-eimDomainName=My Employees,o=My Company, c=US" –w secret
<smfdata
```

The output from the eimadmin command will list a mapping between the UUIDs and the unique identifiers.

Now that you have seen an example of a few of the labels, it is time to look at a comprehensive list of the labels. The following table summarizes the labels associated with object types and command line options associated with these labels.

Table 33. Summary of associated labels

Object Types	Associated Labels	Command Line Options	Descriptions
Domain	DN	-d	Domain distinguished name
	DD	-n	Domain description
	DB	-B	Domain attribute

Table 33. Summary of associated labels (continued)

Object Types	Associated Labels	Command Line Options	Descriptions
Registry	RN	-r	Registry name
	RT	-y	Registry type
	RP	-g	Parent system registry
	RU	-k	Universal resource identifier
	RD	-n	Registry description
	RA	-x	Registry alias (multi-value)
	RZ	-z	Registry alias type
	RB	-B	Registry attribute
Identifier	IU	-i	Unique identifier
	IN	-j	Non-unique identifier (multi-value)
	ID	-n	Identifier description
	II	-o	Identifier information (multi-value)
	UU	-U	Identifier UUID
Association	IU	-i	Unique identifier
	RN	-r	Registry name
	UN	-u	Registry user name
	UC	-E	Certificate file name
	UT	-t	Association type (multi-value)
	UD	-n	User description
	UI	-o	User information (multi-value)
Policy	PT	-t	Policy type
	RS	-r	Source Registry Name
	RG	-T	Target Registry Name
	UN	-u	Target User Name
	UD	-n	User Description
	UI	-o	User information (multi-value)
	UC	-E	Certificate File name
	FT	-G	Certificate Filter Template File Name
	FS	-J	Subject Filter
	FI	-F	Issuer Filter
Access authority	CT	-c	Access authority type
	CU	-q	Access user distinguished name
	CS	-f	Access user type
	RN	-r	Registry name

Rules: Here are the rules for creating a label line:

- Indicate the start of each field by putting a label (from column two of Table 33 on page 130) in the starting column position. Indicate the end of each field by putting a semicolon in the ending column position.

- Separate labels and semicolons with zero or more blanks. (Do not use other white-space characters, such as tabs, because eimadmin interprets them as single blank characters, so a record that visually appears to have correct column positioning might be incorrect for processing.)
- You can specify multi-value labels more than once; for a multi-value label, each value is considered for processing. If you specify more than one value for a single-value label, eimadmin processes only the first value you specify. (See the Description column of Table 33 on page 130 for information about which labels are multi-value.)

Processing differences between command-line options and input files

If you specify information both as command-line options and with an input file, command-line values have priority over input file values. As previously discussed, a single-value option can have only one value. If you specify more than one value, eimadmin processes only the first value. A multi-value option can have more than one value. If you specify more than one value, eimadmin processes all of these values.

If you specify information for a single-value option both within a command and in an input file, eimadmin processes the information on the command rather than that in the file. (This is assuming the value is relevant to the object type and action combination that you specify.)

Note: However, if you specify information for a multi-value option both in a command and in a file, eimadmin processes all the values, processing those in the command first. The eimadmin utility ignores command line options or input file labels that are not appropriate for the object type and action combination that you specify.

The output file

The eimadmin utility generates messages to *stdout* to indicate the following:

- The eimadmin utility version the date and start time, and command options and parameters (as shown in the following example line. See Step 3 on page 133 for the full example)

  ```
  ITY4020 eimadmin (v1) started Mon May 20 10:50:58 2002 with options eimadmin -lI
  ```

- If called to list objects or attributes, the requested information retrieved from the EIM domain
- Whether processing ended normally or stopped due to error
- A summary of processed records (successful and unsuccessful)

The error file

If any errors occur during processing, the eimadmin utility generates error messages to stderr. Failing records are echoed to the output in their entirety. You can correct the failing records and rerun them through the utility. See page Chapter 12, “EIM header file and example,” on page 395 for a sample error file.

Example for adding a list of identifiers to an EIM domain

1. Create a file named ’employees.txt’ containing identity information in a format similar to the following:

```
# Sample eimadmin input file
#
#    User id Birth      Type   Created         First    Nickname  Full       Dept             Hire       Empl
#              date             by              name               name                        date       num
#
```

```
     UN     ;UI       ; UD    ;UI     ;        IN     ; IN       ;IU       ;    II  ;         II       ; ID   ;
021P SMITH   1959-08-01 TSO    ADMIN1  NO  NO  Joseph   Joe       J.C.Smith  DEPTD01  14:20:16 1990-04-11 004321
022P JONES   1968-05-03 TSO    ADMIN1  NO  NO  Robert   Bob       R.Z.Jones  DEPTD01  16:01:57 1988-02-16 001234
023F JONES2  1965-10-15 BATCH  ADMIN4  NO  NO  Robert             R.Z.Jones  DEPTD01  14:12:20 1988-02-16 001234
024P SMITH   1973-11-26        ADMIN3  NO  NO  Joseph   Joe       J.Smith                     1990-04-11 004321
025F BROWN   1970-04-11 TSO    ADMIN3  NO  NO  Charles  Chuck                DEPTD01  09:47:57 1995-01-10 003210

# The following entry was added manually 11/08/01 by ADMINX
026P                           ADMINX          James    Jim       J.Z.Clark      D03           2001-12-22 000012
```

Notes:

a. The exported database can contain information that the eimadmin utility does not use. The two columns with ″NO″ and the column with times between the two II values are such information.

b. There can be only one UN (registry user name), UD (user description), IU (unique identifier), and ID (identifier description).

c. There can be multiple values for UI, IN, and II (user information, non-unique identifier, and identifier information, respectively).

2. Add the identifiers by using the following **eimadmin** command:

```
eimadmin
-aI
-h ldap://my.server
-b "cn=EIM admin,o=My Company, c=US"
-d "ibm-eimDomainName=My Employees,o=My Company, c=US" < employees.txt
> addemployees.out 2> addemployees.err
```

Note: Since the `-w` flag was omitted, the issuer of the **eimadmin** command is prompted for the password.

If the unique identifiers were not previously defined, the output file is the following:

```
ITY4020 eimadmin (v1) started Mon May 20 10:50:58 2002
eimadmin
-aI
-h ldap://my.server
-b "cn=EIM admin,o=My Company, c=US"
-d "ibm-eimDomainName=My Employees,o=My Company, c=US"

ITY4022 6 records processed -- 4  successful; 2 failed.
ITY4021 Processing ended normally.
```

The error file `addemployees.err` contains the following:

```
ITY4030 Service eimAddIdentifier() returned error 117 -- ITY0019 EIM identifier already exists by this name.
ITY4028 Error occurred while processing input line 9.

023F JONES2   1985-10-15 BATCH    ADMIN4    NO   NO   Robert       R.Z.Jones DEPTD01  14:12:20 1988-02-16 001234
ITY4012 Unique identifier not specified.
ITY4028 Error occurred while processing input line 11.

025F BROWN    1990-04-11 TSO      ADMIN3    NO   NO   Charles Chuck          DEPTD01  09:47:57 1995-01-10 003210
```

3. List the identifiers using the same input file by entering the following command:

```
eimadmin
-lI
-h ldap://my.server
-b "cn=EIM admin,o=My Company, c=US"
-d "ibm-eimDomainName=My Employees,o=My Company, c=US" < employees.txt
> listids.out 2> listids.err
```

The file `listids.out` contains output such as the following:

```
ITY4020 eimadmin (v1) started 2001/10/30 at 15:09:00 with options eimadmin -lI
     -hldap://my.server -b "cn=EIM admin,o=My Company, c=US"
    -d "ibm-eimDomainName=My Employees,o=My Company, c=US"
unique identifier: J.C.Smith
 other identifier: J.C.Smith
 other identifier: Joseph
 other identifier: Joe
```

```
      description: 004321
      information: D01
      information: 1990-04-11

unique identifier: R.Z.Jones
 other identifier: R.Z.Jones
 other identifier: Robert
 other identifier: Bob
      description: 001234
      information: D01
      information: 1988-02-16

unique identifier: R.Z.Jones
 other identifier: R.Z.Jones
 other identifier: Robert
 other identifier: Bob
      description: 001234
      information: D01
      information: 1988-02-16

unique identifier: J.Smith
 other identifier: J.Smith
 other identifier: Joseph
 other identifier: Joe
      description: 004321
      information: 1990-04-11

unique identifier: J.Z.W.Clark
 other identifier: J.Z.W.Clark
 other identifier: James
 other identifier: Jim
      description: 000012
      information: D03
      information: 2001-12-22

.
.
.

ITY4022 6 records processed -- 6 successful; 0 failed.
ITY4021 Processing ended normally.
```

While a unique identifier is required for the add action, the eimadmin list action accepts a non-unique identifier when a unique identifier is not provided. The utility searches for entries with the non-unique identifier 'Charles', the first non-unique identifier that appears in the data line. No list output is returned for this line because no matches are found in the domain.

Notes:

a. Notice that the entry for 'R.Z.Jones' is duplicated in the list output. This occurs because there are two data lines with the same unique identifier. The utility processes each line separately, in order of appearance, without recognizing similarities between them.

b. Also notice within each identifier entry that a non-unique value ("other identifier") duplicates the unique identifier value. This is the manner in which the information is stored in LDAP. Do not attempt to remove the duplicate value.

4. You might want to create a number of default registry policies or certificate filter policies using the eim admin input file capability. The following example creates default registry policies for three registry: REG1, REG2, and REG3X509. The following input file, registryPolicies defines the default policies:

```
# default registry policies for REG1, REG2, and REG3X509
# RS - Source registries
# RG - Target registries
# UN - User names in the target registries
# PT - Policy type
RS    ;    RG     ;   UN     ;    PT      ;
REG1       REG2       PUBLIC        REGISTRY
REG1       REG3X509   PUBLIC        REGISTRY
#
REG2       REG1       PUBLIC
REG2       REG3X509   PUBLIC        REGISTRY
#
REG3X509   REG1       PUBLIC     REGISTRY
REG3X509   REG2       PUBLIC        REGISTRY
```

and the following command will cause the updates to be made to the EIM domain:

```
eimadmin -aY -d ibm-eimDomainName=My Employees, o= My Company, c=US -h
ldap://my.server -b "cn=EIM admin,o=My Company,c=US" -w secret <
registryPolicies
```

The REG3X509 registry is an X509 registry so a number of certificate filter policies are defined as well. The following input file, registryFilterPolicies defines the default certificate filter policies:

```
# default certificate filter policies for an X509 registry
# RS - Source registry
# FI - Issuer's filter value
# RG - Target registry
# UN - User name in the target registry
# PT - Policy type
RS     ;     FI                                    ; RG      ;        UN        ;     PT       ;
REG3X509     C=US                                    REG1             PUBLIC          FILTER
REG3X509     C=US                                    REG2             PUBLIC          FILTER
REG3X509     O=A Certificate Authority, L=Internet   REG1             PUBLIC          FILTER
REG3X509     O=A Certificate Authority, L=Internet   REG2             PUBLIC          FILTER
```

and the following command will cause the updates to be made to the EIM domain:

```
eimadmin -aY -d ibm-eimDomainName=My Employees, o= My Company, c=US -h
ldap://my.server -b "cn=EIM admin,o=My Company,c=US" -w secret <
registryFilterPolicies
```

Before the policies take effect, the domain and registries must have policies enabled. The following command will enable policies at the domain level:

```
eimadmin -mD -B POLICY -d ibm-eimDomainName=My Employees, o= My Company, c=US
-h ldap://my.server -b "cn=EIM admin,o=My Company,c=US" -w secret
```

Another input file, registryEnable, can be used to enable policies for each registry in the domain:

```
# enable the policies in the registries
# RN - Registry name
# RB - Registry attribute
RN   ;     RB    ;
REG1       POLICY
REG2       POLICY
REG3X509   POLICY
```

and the following command will make the udpates to the registries:

```
eimadmin -mR -d ibm-eimDomainName=My Employees, o= My Company, c=US -h
ldap://my.server -b "cn=EIM admin,o=My Company,c=US" -w secret < registryEnable
```

5. Associations between identifiers and certificates can be created by using the input file capability of eimadmin. In this example, the current directory contains three files with the certificates of three users. The input file, certMappings contains the information required to define the relationships with EIM identifiers:

```
# Mappings between identifiers and certificates in files
# IU - Unique identifier
# UC - Certificate file name containing the user's certificate
IU    ;       UC   ;
John Day    ./JohnDayCert
Jill Jack   ./JillJackCert
Jane Day    ./JaneDayCert
```

By issuing the following command, the source associations between the users name in the certificates and their EIM identifiers will be added to the domain:

```
eimadmin -aA -r REG3X509 -s SOURCE -d ibm-eimDomainName=My Employees, o= My
Company, c=US -h ldap://my.server -b "cn=EIM admin,o=My Company,c=US" -w secret <
certMappings
```

Note: In order for this to work, the registry must have a registry type of X509.

Using eimadmin with the tabular output of SMF Unload

The audit records cut by EIM contain the UUID of identifiers and policies. eimadmin can be used to get the friendly name associated with the UUID.

First, generate the tabular output using SMF Unload. Then edit the tabular output so that the 'UU' label appears above the UUID. For this example, assume the data is in the HFS file smfdata.

Then run the following eimadmin command:

```
eimadmin -lI -h ldap://my.server -b "cn=EIM admin,o=My Company, c=US"
-d "ibm-eimDomainName=My Employees,o=My Company,c=US" –w secret
<smfdata
```

The output from the eimadmin command will list a mapping between the UUIDs and the unique identifiers.

Chapter 10. EIM Auditing

Auditing is an important part of keeping your enterprise secure. Applications which allow you to use user ID mappings must be kept secure to locate an end user's identity on a server system and trust that the end user has been correctly authenticated at the client. This means that the data mapped via EIM must be kept accurate and secure. To facilitate this, EIM allows you to perform auditing functions on select EIM APIs.

Auditing EIM events

Existing RACF commands are used to identify what events to log and when to log them. The EIM audit log records are written to SMF as type-83 subtype 2 records. Authentication and authorization failures are displayed on the security console and stored in the caller's job log. The SMF type-83 log records containing EIM events can be unloaded using the RACF SMF Data Unload utility for further analysis by auditing tools.

Categories of EIM events

EIM events are divided into three groups of resources:

- EIM Connect (including disconnects)
- EIM Admin
- EIM Lookup

Each group of events is qualified by whether or not the specific API was successful, if it failed because of an authentication or authorization failure, or if the data was not found. The controls a security administrator or auditor can use to implement a security policy are defined using grouping of events as well. Each group of events is controlled by a resource in the RAUDITX class. A profile in the RAUDITX class can be defined with the appropriate auditing options. The resources are:

- EIM.*domain name*.CONNECT
- EIM.*domain name*.ADMIN
- EIM.*domain name*.LOOKUP

Some events always cause a log record to be written. For example, attempts to connect with an EIM domain controller that fail because the bind user could not be authenticated.

Table 34. Events which are always logged

Resource in the RAUDITX class	Events
EIM.domain name.CONNECT	Connects, disconnects with an EIM domain controller
EIM.domain name.ADMIN	All loggable EIM administration tasks, such as creating, deleting, modifying, or listing domains, identifiers, associations, and policies
EIM.domain name.LOOKUP	EIM lookup operations that retrieve mappings from an EIM domain, such as eimGetTargetFromSource, eimGetTargetFromIdentifier, or eimGetAssociatedIdentifiers

Profiles can be defined in the RAUDITX class that can apply to all events or a subset.

Table 35. Covering profiles in the RAUDITX class and descriptions

EIM.*	All EIM events, all domains
EIM.domain.name.*	All EIM events in a particular domain
EIM.domain name.CONNECT	All attempts to bind/unbind with a particular domain; bind failures are always logged
EIM.domain name.ADMIN	All admin events with a particular domain
EIM.domain name.LOOKUP	All lookup events with a particular domain

The domain name used in the resource name and RACF profile is the same as the EIM domain name with the following differences:

- all blanks, commas, parenthesis, semicolons, asterisks, percent signs, and ampersands are removed
- the compressed name is truncated if it exceeds 234 characters
- lowercase characters are treated the same as uppercase

Generics must be active for the RAUDITX class before the profiles EIM.* and EIM.domain name.* profiles are created. Use the SETROPTS GENERIC(RAUDITX) command to activate generics.

The log records that are generated can be extracted from the SMF dataset by using the RACF SMF Data Unload tool to convert the logged data into a format that can be analyzed using a relational database manager or a tool like DFSORT ICETOOL or into an XML document.

Each EIM API belongs to one of these resource names. The following table lists the APIs and the resource name they belong to.

Table 36. EIM Event Categories

EIM API	**Resource Name**			
	EIM. domain name. CONNECT	**EIM. domain name. ADMIN**	**EIM. domain name. LOOKUP**	**No logging performed**
eimAddAccess		X		
eimAddApplicationRegistry		X		
eimAddAssociation		X		
eimAddIdentifier		X		
eimAddPolicyAssociation		X		
eimAddPolicyFilter		X		
eimAddSystemRegistry		X		
eimChangeDomain		X		
eimChangeIdentifier		X		
eimChangeRegistry		X		
eimChangeRegistryAlias		X		
eimChangeRegistryUser		X		
eimConnect	X*			

Table 36. EIM Event Categories (continued)

eimConnectToMaster	X*			
eimCreateDomain		X		
eimCreateHandle				X
eimDeleteDomain		X		
eimDestroyHandle	X***			
eimErr2String				X
eimFormatPolicyFilter				X
eimFormatUserIdentity				X
eimGetAssociatedIdentifiers			X**	
eimGetAttribute				X
eimGetRegistryNameFromAlias				X
eimGetTargetFromIdentifier			X**	
eimGetTargetFromSource			X**	
eimGetVersion				X
eimListAccess				X
eimListAssociations				X
eimListDomains *				X
eimListIdentifiers				X
eimListPolicyFilters				X
eimListRegistries				X
eimListRegistryAliases				X
eimListRegistryAssociations				X
eimListRegistryUsers				X
eimListUserAccess				X
eimQueryAccess				X
eimRetrieveConfiguation				X
eimRemoveAccess		X		
eimRemoveAssociation		X		
eimRemoveIdentifier		X		
eimRemovePolicyAssociation		X		
eimRemovePolicyFilter		X		
eimRemoveRegistry		X		
eimSetAttribute				X
eimSetConfigurationExt				X

Notes:

1. * Authentication failures are always logged for the EIM connect APIs.
2. ** Not found conditions are logged in addition to success and failure for the lookup APIs.
3. *** eimDestroyHandle events are only logged after a successful connect.

How events are audited

Events can be audited by one of the following methods:

- setting SETROPTS LOGOPTIONS
- enabling auditing for resources in the RAUDITX class
- enabling auditing for a specific user

There are a number of setup steps that must be performed by the security administrator before EIM events are logged. The user id of the EIM application must be given the authority to use the auditing service.

```
RDEFINE FACILITY IRR.RAUDITX UACC(NONE)
PERMIT IRR.RAUDITX CLASS(FACILITY) ID(userid) ACCESS(READ)
SETR CLASSACT(FACILITY)
```

With this basic configuration, failures to connect to the EIM domain because of bind failures due to bad passwords are logged.

The security administrator or auditor can issue additional commands to control logging:

- Activate (or deactivate) logging for all EIM events involving all EIM domains by using SETROPTS LOGOPTIONS for the RAUDITX class. The RAUDITX class must be active for the SETROPTS LOGOPTIONS settings to take effect:

  ```
  SETROPTS LOGOPTIONS(auditing-level(RAUDITX))
  ```

  where the auditing-level can be ALWAYS, NEVER, SUCCESSES, FAILURES, or DEFAULT.
- Activate or deactivate logging for a subset of EIM events by using SETROPTS LOGOPTIONS(DEFAULTS(RAUDITX). The commands are:

  ```
  SETROPTS LOGOPTIONS(DEFAULT(RAUDITX))
  RDEFINE RAUDITX profile-name UACC(NONE)
  PERMIT profile-name CLASS(RAUDITX) ID(user or group) ACCESS(READ)
  SETROPTS CLASSACT(RAUDITX)
  SETROPTS RACLIST(RAUDITX) or SETROPTS RACLIST(RAUDITX) REFRESH
  ```
- Activate or deactivate logging for a particular user:

  ```
  ALTUSER user id UAUDIT
  ```

Generic profile names may be used when generics are activated for the RAUDITX class:

```
SETR GENERIC(RAUDITX)
```

This is one approach for activating logging of EIM events. Consult with your security administrator for other options. See *z/OS Security Server RACF Command Language Reference* and *z/OS Security Server RACF Security Administrator's Guide* for more information on the RACF commands, general resources, RACLIST processing, and generic profile processing.

SETROPTS LOGOPTIONS activates or deactivates event logging for all resources in a class. The auditor must have the RACF authority required by the SETROPTS command. See *z/OS Security Server RACF Command Language Reference* for more details on the authorities required to issue commands.

Following are the combinations of SETROPTS commands that can be used to enable or disable auditing for all EIM events in all EIM domains:

Table 37. SETROPTS options for audit enablement

Command	Results	Explanation
SETROPTS LOGOPTIONS(NEVER(RAUDITX))	No log records except authentication connection failures	No log records are written for any domain except for connection failures.
SETROPTS LOGOPTIONS(ALWAYS(RAUDITX))	Successful or failure authorization or authentication	Log records are written for all EIM events in all domains.
SETROPTS LOGOPTIONS(SUCCESSES(RAUDITX))	Successful and connection failure records	Success records are written for all EIM events in all domains. **Note:** This applies to all resources in the RAUDITX class.
SETROPTS LOGOPTIONS(FAILURES(RAUDITX))	Failure records	Failure records are written for all EIM events in all domains.
SETROPTS LOGOPTIONS(DEFAULTS(RAUDITX))	None.	The decision to log is determined by profiles in the RAUDITX class.

Note: When you enable auditing using SETROPTS, you are enabling auditing for all applications that use the RAUDITX class, not just EIM.

Auditing can also be enabled and controlled for a specific profile. In this case, the audit settings set by the profile owner or auditor are used to determine when and what to audit. Therefore, the profile owner has more direct control over auditing of their profiles.

Table 38. Enabling EIM Auditing Using Profiles

Commands	Results	Explanation
RDEFINE RAUDITX EIM.* AUDIT(...) or RALTER RAUDITX EIM.* GLOBALAUDIT(...)	Success and/or failure	Log records are written for all EIM events for all users in all domains.
RDEFINE RAUDITX EIM.*.CONNECT AUDIT(...) or RALTER RAUDITX EIM.*.CONNECT GLOBALAUDIT(...)	Connect success and failure. Disconnect success.	Log records are written only for connect / disconnects for all users in all domains.
RDEFINE RAUDITX EIM.*.ADMIN AUDIT(...) or RALTER RAUDITX EIM.*.ADMIN GLOBALAUDIT(...)	Admin success and/or failure.	Log records are written for any administration related event in any domain. **Note:** Only failures are written for list events or retrieve events.
RDEFINE RAUDITX EIM.*.LOOKUP AUDIT(...) or RALTER RAUDITX EIM.*.LOOKUP GLOBALAUDIT(...)	Lookup success and/or failure.	Log records are written for any lookup related event in any domain.

Generics must be active for the RAUDITX class before the profiles EIM.*, EIM.*.CONNECT, EIM.*.ADMIN, or EIM.*.LOOKUP are created. Use the SETROPTS GENERIC(RAUDITX) command to enable generics.

Additionally, an auditor can activate logging of events for a specific user by issuing the following:

```
ALTUSER userid UAUDIT
```

In this case, all events for this user are audited, including non-EIM events.

Regardless of the level at which auditing is enabled, the following events are always audited:

EIM.domain name.CONNECT
All connection authentication failures with an EIM domain controller

What goes into an audit record

For each audited event, the following information is captured:

- information about the event:
  - general event code
  - the event code qualifier (success, bind failure, not found)
  - the name of the EIM API that logged the event (C/C++ API name)
- the EIM domain URL
- the user
  - the bind user name
  - the current RACF user ID
- the input parameters to the EIM API and in some cases additional information about the event, such as the UUID for the EIM identifier
- the results of the event in some cases such as the user IDs returned by an eimGetTargetFromSource
- the reasons the event was logged (EIM required logging of authentication failures, SETROPTS LOGOPTION settings, the audit settings on the RACF profile that covers the resource name representing the event)
- a link value which is used to identify multiple log records that were generated for a single event

Working with audit records

The EIM events are logged in an SMF dataset as type 83 subtype 2 records. The log record is a mixture of binary and EBCDIC data. The general format of SMF Type 83 records is described in *z/OS Security Server RACF Macros and Interfaces*, in the chapter ″SMF records″, section ″Record type 83: Security Events.″

You can use the RACF SMF Unload utility to reformat the EIM SMF type 83 subtype 2 records for easier analysis. The audit records can be used to identify access violations or determine patterns of usage from your users. These audit records can be in the following different forms:

- A tabular format, suitable for import to a relational database manager.
- eXtensible Markup Language (XML) documents

Information on how to use the RACF SMF Unload utility can be found in *z/OS Security Server RACF Auditor's Guide*. This book describes how the reformatted data can be further processed by DB2, DFSORT, and XML applications.

The next sections gives details on the different formats of the EIM event data:

- The layout of the EIM data contained in an SMF type 83 subtype 2 record
- The schema definition for the EIM XML elements
- Field names and column positions of the EIM data in the tabular output and sample DB2 statements for loading the EIM data into DB2

The SMF Record Type 83 subtype 2 records

When auditing is enabled for EIM events, SMF record type 83 subtype 2 records are recorded in the SMF dataset. Each logged EIM event has a unique event code with a corresponding event code qualifier, or value that indicates if the event succeeded, failed because the end user was not authorized, or if the requested data was not found. The event code and event code qualifiers are described in the following table.

Table 39. EIM event codes

Event	Command / Service	Code Qualifier Dec (Hex)	Description	Relocate type sections (possible SMF83TP2 / SMF83DA2 values)
1(1)	EIM Connection Events	0(0)	Successful connect to the domain controller or a disconnection from the domain controller	Common relocates, 100-105
		3(3)	Not authorized to connect to the domain controller	
2(2)	EIM Lookup Events	0 (0)	Successful request	Common relocates, 100-103, 106-113, 119, 127, 131, 136
		1(1)	Insufficient authority to retrieve EIM data	
		2(2)	Mapping not found or the user was not authorized to access the EIM data	

Table 39. EIM event codes (continued)

Event	Command / Service	Code Qualifier Dec (Hex)	Description	Relocate type sections (possible SMF83TP2 / SMF83DA2 values)
3(3)	EIM Administrative Events – Domain, Registry, Access	0 (0)	Successful request	Common relocates, 100-105, 117-120, 122, 125, 128-134
		1(1)	Insufficient authority to modify the EIM domain or retrieve information from the domain	
		3(3)	Not authorized to connect to the EIM domain controller	
4(4)	EIM Administrative Events – Identifiers, Associations, Policies	0(0)	Successful request	Common relocates, 100-103, 106-113, 119, 120, 123-127 131, 135, 136, 137
		1(1)	Insufficient authority to modify the EIM domain or retrieve information from the domain	

For more information on SMF records and the relocates that are common to all SMF Type 83 subtype 2 and above records, see *z/OS Security Server RACF Macros and Interfaces*, section ″Record type 83: Security Events.″

The following are the EIM specific extended relocates:

Table 40. EIM extended relocates

Data Type (SMF83TP2)		Max Data Length (SMF83DL2)		Format	Audited by Event Code	Description
Dec	Hex	Dec	Hex			
100	64	128	80	EBCDIC	1, 2, 3, 4	EIM API Name
101	65	512	200	EBCDIC	1, 2, 3, 4	Domain URL
102	66	64	40	EBCDIC	1, 2, 3, 4	Connection Type - SIMPLE, SIMPLE AND CRAM - MD5, SIMPLE CRAM_MD5 OPTIONAL, KERBEROS, SSL CLIENT AUTH
103	67	512	200	EBCDIC	1, 2, 3, 4	Bind user
104	68	256	100	EBCDIC	1, 3	Certificate Label
105	69	256	100	EBCDIC	1, 3	Key Ring

Table 40. EIM extended relocates (continued)

106	6A	256	100	EBCDIC	2, 4	Registry name - Source
107	6B	256	100	EBCDIC	2, 4	Registry user name - Source
108	6C	36	24	EBCDIC	2, 4	Identifier UUID (fixed length)
109	6D	256	100	EBCDIC	2, 4	Identifier unique name
110	6E	256	100	EBCDIC	2, 4	Identifier alias
111	6F	256	100	EBCDIC	2, 4	Registry name - Target
112	70	256	100	EBCDIC	2, 4	Registry user additional info
113	71	256	100	EBCDIC	2, 4	Registry user name – Target
114	72	--	--	--	--	Reserved
115	73	--	--	--	--	Reserved
116	74	--	--	--	--	Reserved
117	75	64	40	EBCDIC	3	Access type - EIM_ACCESS_ADMIN, EIM_ACCESS_REG_ADMIN, EIM_ACCESS_REGISTRY, EIM_ACCESS_IDENTIFIER_ADMIN, EIM_ACCESS_MAPPING_LOOKUP
118	76	256	100	EBCDIC	3	Access user
119	77	64	40	EBCDIC	2, 3, 4	Association or Policy Type - EIM_ADMIN, EIM_ALL_ASSOC, EIM_ALL_POLICY_ASSOC, EIM_CERT_FILTER_POLICY, EIM_DEFAULT_DOMAIN_POLICY, EIM_DEFAULT_REG_POLICY, EIM_SOURCE, EIM_SOURCE_AND_TARGET, EIM_TARGET
120	78	64	40	EBCDIC	3, 4	Change Type - EIM_ADD, EIM_CHG, EIM_ENABLE, EIM_DISABLE, EIM_RMV
121	79	--	--	--	--	Reserved
122	7A	256	100	EBCDIC	3	Domain Description
123	7B	256	100	EBCDIC	4	Identifier additional information
124	7C	256	100	EBCDIC	4	Identifier description
125	7D	256	100	EBCDIC	3, 4	Options (multi-valued) - EIM_FAIL, EIM_GEN_UNIQUE, EIM_REGISTRY_MAPPING_LOOKUP, EIM_REGISTRY_POLICY_ ASSOCIATIONS
126	7E	64	40	EBCDIC	4	Policy Filter Type - EIM_ALL_FILTERS, EIM_CERTIFICATE_FILTER
127	7F	512	200	EBCDIC	2, 4	Policy Filter Value
128	80	64	40	EBCDIC	3	Registry alias type
129	81	256	100	EBCDIC	3	Registry alias value
130	82	256	100	EBCDIC	3	Registry description
131	83	256	100	EBCDIC	2, 3, 4	Registry name or Registry name - ADMIN

Table 40. EIM extended relocates (continued)

132	84	256	100	EBCDIC	3	Registry name – System
133	85	64	40	EBCDIC	3	OID for the Registry type
134	86	256	100	EBCDIC	3	Registry URI
135	87	256	100	EBCDIC	2, 4	Registry user description
136	88	256	100	EBCDIC	4	Registry user name
137	89	256	100	EBCDIC	4	Identifier alias attribute value

The XML output from the RACF SMF Unload Utility

The EIM XML elements generated by the RACF SMF Unload utility for an EIM event are described by the XML schema document IRREIMSC which can be found in SYS1.SAMPLIB.

The following is a copy of the EIM schema document.

```
<?xml version="1.0" encoding="ebcdic-cp-us" ?>
<xs:schema targetNamespace="http://www.ibm.com/xmlns/zOS/EIMSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.ibm.com/xmlns/zOS/EIMSchema">

  <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
  <!--                                                                     -->
  <!-- PROPRIETARY STATEMENT                                               -->
  <!--                                                                     -->
  <!--                                                                     -->
  <!-- LICENSED MATERIALS - PROPERTY OF IBM                                -->
  <!-- THIS MACRO IS "RESTRICTED MATERIALS OF IBM"                         -->
  <!-- 5637-A01 (C) COPYRIGHT IBM CORP. 2005                               -->
  <!--                                                                     -->
  <!-- STATUS= HRF7720                                                     -->
  <!--                                                                     -->
  <!-- END_OF_PROPRIETARY_STATEMENT                                        -->
  <!--                                                                     -->
  <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
  <!--                                                                     -->
  <!--*01* EXTERNAL CLASSIFICATION: OTHER                                 -->
  <!--*01* END OF EXTERNAL CLASSIFICATION:                                -->
  <!--                                                                     -->
  <!--  This SAMPLIB member is only an example. The value                  -->
  <!--  on each statement is not necessarily an IBM-recommended            -->
  <!--  value.  Installations may use this member to validate XML          -->
  <!--  documents produced by the RACF SMF Data Unload Utility             -->
  <!--                                                                     -->
  <!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
  <!--                                                                     -->
  <!-- Name:        HITSCHEM                                               -->
  <!--                                                                     -->
  <!-- Description: Define the EIM XML grammar used by the XML             -->
  <!--              instance documents produced by the RACF SMF Data       -->
  <!--              Unload Utility (IRRADU00).                             -->
  <!--                                                                     -->
  <!-- Operation:   This is an XML schema document that defines the        -->
  <!--              XML tag language used by the RACF SMF Data             -->
  <!--              Unload Utility(IRRADU00) XML instance documents        -->
  <!--              containing EIM events.                                 -->
  <!--                                                                     -->
  <!--                                                                     -->
  <!-- CHANGE ACTIVITY:                                                    -->
  <!--  $L0=EIMAD    HRF7720 040211 PDMKL1 EIM Auditing          @L0A-->
  <!--  $P1=EIMAD    HRF7720 040525 PDMKL1 MG03949               @P1A-->
  <!--  $P2=EIMAD    HRF7720 040608 PDMKL1 MG03969               @P2A-->
```

```
<!--  $P3=EIMAD    HRF7720 040608 PDMKL1 MG04010              @P3A-->
<!--  $P4=EIMAD    HRF7720 040625 PDMKL1 MG04222              @P4A-->
<!--  $P5=EIMAD    HRF7720 040727 PDMKL1 MG04417              @P5A-->
<!--  $P5=EIMAD    HRF7720 040729 PDMKL1 MG04444              @P6A-->
<!--                                                             -->
<!--                                                             -->
<!-- CHANGE DESCRIPTIONS:                                        -->
<!--  A000000-999999                                          @L0A-->
<!--  C - Updated the enumerated values for t_accessType,     @P1A-->
<!--      t_assocPolicyType, t_changeType                     @P1A-->
<!--      t_connectionType, t_policyFilterType, and           @P1A-->
<!--      t_options                                           @P1A-->
<!--  A - Added identAliasat element.                         @P2A-->
<!--  D - Delete regLookups, domainPol, and regPol elements   @P3A-->
<!--  C - Shortened records so they fit into SYS1.SAMPLIB     @P4A-->
<!--  D - Deleted last line which contained an invalid char   @P5A-->
<!--  C - Updated t_connectionType, t_policyFilterTypes,      @P6A-->
<!--      identUuid, regAliastype elements.                   @P6A-->
<!--                                                             -->
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

<!--                                                             -->
<!-- EIM base types                                              -->
<!--                                                             -->

<xs:simpleType name="t_connectionType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="SIMPLE"/>
    <xs:enumeration value="SIMPLE AND CRAM_MD5"/>
    <xs:enumeration value="PROTECT CRAM_MD5 OPTIONAL"/>
    <xs:enumeration value="KERBEROS"/>
    <xs:enumeration value="SSL CLIENT AUTH"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="t_accessType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="EIM_ACCESS_ADMIN"/>
    <xs:enumeration value="EIM_ACCESS_REG_ADMIN"/>
    <xs:enumeration value="EIM_ACCESS_REGISTRY"/>
    <xs:enumeration value="EIM_ACCESS_IDENTIFIER_ADMIN"/>
    <xs:enumeration value="EIM_ACCESS_MAPPING_LOOKUP"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="t_assocPolicyType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="EIM_ADMIN"/>
    <xs:enumeration value="EIM_ALL_ASSOC"/>
    <xs:enumeration value="EIM_ALL_POLICY_ASSOC"/>
    <xs:enumeration value="EIM_CERT_FILTER_POLICY"/>
    <xs:enumeration value="EIM_DEFAULT_DOMAIN_POLICY"/>
    <xs:enumeration value="EIM_DEFAULT_REG_POLICY"/>
    <xs:enumeration value="EIM_SOURCE"/>
    <xs:enumeration value="EIM_SOURCE_AND_TARGET"/>
    <xs:enumeration value="EIM_TARGET"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="t_changeType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="EIM_ADD"/>
    <xs:enumeration value="EIM_CHG"/>
    <xs:enumeration value="EIM_DISABLE"/>
    <xs:enumeration value="EIM_ENABLE"/>
    <xs:enumeration value="EIM_RMV"/>
  </xs:restriction>
```

```
</xs:simpleType>

<xs:simpleType name="t_options">
  <xs:restriction base="xs:token">
    <xs:enumeration value="EIM_FAIL"/>
    <xs:enumeration value="EIM_GEN_UNIQUE"/>
    <xs:enumeration value="EIM_REGISTRY_MAPPING_LOOKUP"/>
    <xs:enumeration value="EIM_REGISTRY_POLICY_ASSOCIATIONS"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="t_policyFilterType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="EIM_CERTIFICATE_FILTER"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="t_string">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="t_string36">
  <xs:restriction base="xs:string">
    <xs:length value="36"/>
  </xs:restriction>
</xs:simpleType>

<!--                                                                -->
<!-- EIM specific elements.                                         -->
<!--                                                                -->
<xs:element name="accessType"   type="t_accessType"               />
<xs:element name="accessUser"   type="t_string"                   />
<xs:element name="api"          type="t_string"                   />
<xs:element name="assocpolType" type="t_assocPolicyType"          />
<xs:element name="bindUser"     type="t_string"                   />
<xs:element name="certLabel"    type="t_string"                   />
<xs:element name="changeType"   type="t_changeType"               />
<xs:element name="connectType"  type="t_connectionType"           />
<xs:element name="domainDesc"   type="t_string"                   />
<xs:element name="domainUrl"    type="t_string"                   />
<xs:element name="identAlias"   type="t_string"                   />
<xs:element name="identAliasat" type="t_string"                   />
<xs:element name="identDesc"    type="t_string"                   />
<xs:element name="identInfo"    type="t_string"                   />
<xs:element name="identUnique"  type="t_string"                   />
<xs:element name="identUuid"    type="t_string36"                 />
<xs:element name="keyRing"      type="t_string"                   />
<xs:element name="options"      type="t_options"                  />
<xs:element name="polFilt"      type="t_string"                   />
<xs:element name="polFiltType"  type="t_policyFilterType"         />
<xs:element name="regAlias"     type="t_string"                   />
<xs:element name="regAliastype" type="t_string"                   />
<xs:element name="regDesc"      type="t_string"                   />
<xs:element name="regName"      type="t_string"                   />
<xs:element name="regSrc"       type="t_string"                   />
<xs:element name="regSystem"    type="t_string"                   />
<xs:element name="regTgt"       type="t_string"                   />
<xs:element name="regType"      type="t_string"                   />
<xs:element name="regUri"       type="t_string"                   />
<xs:element name="regUserDesc"  type="t_string"                   />
<xs:element name="regUserInfo"  type="t_string"                   />
<xs:element name="regUserName"  type="t_string"                   />
```

```
  <xs:element name="regUserSrc"   type="t_string"                    />
  <xs:element name="regUserTgt"   type="t_string"                    />

</xs:schema>
```

The tabular output from the RACF SMF Unload utility

A general description of the format of the tabular records created by the RACF SMF Unload Utility can be found in *z/OS Security Server RACF Macros and Interfaces*, in the chapter ″The format of the unloaded SMF type 83 data″.

For EIM events, SMF Unload reserves two spaces between fields that appear after the header information. When the length of a value in the SMF type 83 subtype 2 record exceeds the available field length in the tabular output, SMF Unload will insert a ″+″ in the character position following indicating more data is available. This additional data can be viewed by reading the logged SMF record or by creating an XML document from the logged SMF record.

The following table describes the format of the header portion of the record. RACF constructs the header portion of the record from the SMF record. Because each of the SMF record types that SMF Unload processes contain different data, some fields of the header portion of the unloaded SMF record contain blanks. When working with the sample DB2 statements for loading the data into DB2, the <col_id> string is replaced by the column identifier for each field created. For EIM fields, the <col_id> may have the values:

Table 41. <col_id> values

<col_id>	**Description**
EIMC	EIM connection events
EIML	EIM lookup events
EIMD	EIM administrative events involving changes to EIM domains, registries, and user access to data in an EIM domain
EIMI	EIM administrative events involving changes to identifiers, associations, and policies

Table 42. Common information in the SMF Type 83 Subtype 2 records

Field Name	**Type**	**Length**	**Start**	**End**	**Comments**
<col_id>_EVENT_TYPE	char	8	1	8	The type of the event. Set to *CONNECT, *LOOKUP, *ADMIN, or *ADMIN2
<col_id>__EVENT_QUAL	char	8	10	17	The type of a qualification of the type of event that is being described. Valid values are shown in the tables that accompany each of the record extension descriptions.
<col_id>__TIME_WRITTEN	Time	8	19	26	Time that the record was written to SMF.
<col_id>_DATE_WRITTEN	Date	10	28	37	Date that the record was written to SMF.

Table 42. Common information in the SMF Type 83 Subtype 2 records (continued)

Field Name	Type	Length	Start	End	Comments
<col_id>_SYSTEM_SMFID	Char	4	39	42	SMF system ID of the system from which the record originates.
<col_id>_RESERVED_01	Char	16	44	59	Reserved.
<col_id>_VIOLATION	Yes/No	4	61	64	Does this record represent a violation?
<col_id>_USER_NDFND	Yes/No	4	66	69	Was this user not defined to RACF?
<col_id>_USER_WARNING	Yes/No	4	71	74	Was this record created because of a WARNING?
<col_id>_EVT_USER_ID	Char	8	76	83	User ID associated with the event.
<col_id>_EVT_GRP_ID	Char	8	85	92	Group name associated with the event.
<col_id>_AUTH_NORMAL	Yes/No	4	94	97	Was normal authority checking a reason for access being allowed?
<col_id>_AUTH_SPECIAL	Yes/No	4	99	102	Was special authority checking a reason for access being allowed?
<col_id>_AUTH_OPER	Yes/No	4	104	107	Was operations checking a reason for access being allowed?
<col_id>_AUTH_AUDIT	Yes/No	4	109	112	Was auditor authority checking a reason for access being allowed?
<col_id>_AUTH_EXIT	Yes/No	4	114	117	Was exit checking a reason for access being allowed?
<col_id>_AUTH_FAILSFT	Yes/No	4	119	122	Was failsoft checking a reason for access being allowed?
<col_id>_AUTH_BYPASS	Yes/No	4	124	127	Was the use of the user ID *BYPASS* a reason for access being allowed?
<col_id>_AUTH_TRUSTED	Yes/No	4	129	132	Was trusted authority checking a reason for access being allowed?
<col_id>_LOG_CLASS	Yes/No	4	134	137	Was SETR AUDIT(class) checking a reason for this event to be recorded?
<col_id>_LOG_USER	Yes/No	4	139	142	Was auditing requested for this user?

Table 42. Common information in the SMF Type 83 Subtype 2 records (continued)

Field Name	Type	Length	Start	End	Comments
<col_id>_LOG_SPECIAL	Yes/No	4	144	147	Was auditing requested for access granted due to the SPECIAL privilege?
<col_id>_LOG_ACCESS	Yes/No	4	149	152	Did the profile indicate audit, or did FAILSOFT processing allow access, or did the RACHECK exit indicate auditing?
<col_id>_LOG_RACINIT	Yes/No	4	154	157	Did the RACINIT fail?
<col_id>_LOG_ALWAYS	Yes/No	4	159	162	Is this command always audited?
<col_id>_LOG_CMDVIOL	Yes/No	4	164	167	Was this event audited due to CMDVIOL?
<col_id>_LOG_GLOBAL	Yes/No	4	169	172	Was this event audited due to GLOBALAUDIT?
<col_id>_TERM_LEVEL	Integer	3	174	176	The terminal level associated with this audit record.
<col_id>_BACKOUT_FAIL	Yes/No	4	178	181	Did RACF fail in backing out the data?
<col_id>_PROF_SAME	Yes/No	4	183	186	Was the profile the same at the end of this event?
<col_id>_TERM	Char	8	188	195	The terminal associated with the event.
<col_id>_JOB_NAME	Char	8	197	204	The job name associated with the event.
<col_id>_READ_TIME	Time	8	206	213	The time that the job entered the system.
<col_id>_READ_DATE	Date	10	215	224	The date that the job entered the system.
<col_id>_SMF_USER_ID	Char	8	226	233	User ID from SMF common area. This value is managed by SMF processing exits.
<col_id>_LOG_LEVEL	Yes/No	4	235	238	Was this event audited due to SECLEVEL auditing?
<col_id>_LOG_LOGOPT	Yes/No	4	240	243	Was this event audited due to SETR LOGOPTIONS auditing?

Table 42. Common information in the SMF Type 83 Subtype 2 records (continued)

Field Name	Type	Length	Start	End	Comments
<col_id>_LOG_SECL	Yes/No	4	245	248	Was this event audited due to SETR SECLABELAUDIT auditing?
<col_id>_LOG_COMPATM	Yes/No	4	250	253	Was this event audited due to SETR COMPATMODE auditing?
<col_id>_LOG_APPLAUD	Yes/No	4	255	258	Was this event audited due to SETR APPLAUDIT?
<col_id>_USR_SECL	Char	8	260	267	The SECLABEL associated with this user.
<col_id>_LOG_VMEVENT	Yes/No	4	269	272	Was this event audited due to VMEVENT auditing?
<col_id>_LOG_NONOMVS	Yes/No	4	274	277	Did this user try to use z/OS UNIX without being defined as a z/OS UNIX user (that is, is the user's OMVS segment in the RACF database missing)?
<col_id>_LOG_OMVSNPRV	Yes/No	4	279	282	The service that was requested requires that the user be the z/OS UNIX superuser.
<col_id>_AUTH_OMVSSU	Yes/No	4	284	287	Was the z/OS UNIX superuser authority used to grant the request?
<col_id>_AUTH_OMVSSYS	Yes/No	4	289	292	Was the request granted because the requester was z/OS UNIX itself?
<col_id>_RACF_VERSION	Char	4	294	297	The version of RACF on the system which audited the event
<col_id>_SRVR_USER_ID	Char	8	299	306	Identifier of the address space user associated with this event (jobname is used if the user is not defined to RACF)
<col_id>_SRVR_GRP_ID	Char	8	308	315	Group to which the address space user was connected (stepname is used if the user is not defined to RACF)

Table 42. Common information in the SMF Type 83 Subtype 2 records (continued)

Field Name	Type	Length	Start	End	Comments
<col_id>_PROD_ID	Char	8	317	324	Short name for the product logging the event.
<col_id>_LOG_RAUDITX	Yes/No	4	326	329	Did the caller of R_auditx require logging of this event?
<col_id>_X500_SUBJECT	Char	255	331	585	Subject's name associated with this event.
<col_id>_X500_ISSUER	Char	255	588	842	Issuer's name associated with this event
<col_id>_RES_NAME	Char	246	845	1090	Resource name
<col_id>_CLASS	Char	8	1093	1100	Class name
<col_id>_NAME	Char	246	1103	1348	Profile name
<col_id>_PROD_FMID	Char	7	1351	1357	The version of the product or component which detected the event
<col_id>_PROD_NAME	Char	255	1360	1614	The name of the product or component which detected the event
<col_id>_LOGSTR	Char	255	1617	1871	Log string
<col_id>_EVENT_LINK	Char	16	1874	1889	Value used to link several SMF records to the same event.
<col_id>_RESERVED_02	Char	1105	1892	2997	Reserved.

The data following the header in the tabular record varies according to the EIM event that was recorded. There are four EIM event types:

Event Code from SMF type 83 subtype 2 records	Tabular output Event Type strings
1	*CONNECT
2	*LOOKUP
3	*ADMIN1
4	*ADMIN2

The following tables describe the data that may appear in the tabular record for each of the EIM event types and the event qualifiers for each event type.

*Table 43. Event-specific fields for EIM connection events (EIMC_EVENT_TYPE is "*CONNECT")*

Field Name	Type	Length	Position		Comments
			Start	End	
EIMC_API	Char	128	3000	3127	EIM API Name
EIMC_DOMAIN_URL	Char	128	3130	3257	Domain URL

*Table 43. Event-specific fields for EIM connection events (EIMC_EVENT_TYPE is "*CONNECT") (continued)*

Field Name	Type	Length	Position		Comments
			Start	End	
EIMC_CONNECT_TYPE	Char	32	3260	3291	Connection type - SIMPLE, SIMPLE AND CRAM-MD5, SIMPLE CRAM_MD5 OPTIONAL, KERBEROS, SSL CLIENT AUTH
EIMC_BIND_USER	Char	512	3294	3805	Bind user
EIMC_CERT_LABEL	Char	128	3808	3935	Certificate Label
EIMC_KEY_RING	Char	128	3938	4065	Key Ring

The event qualifiers (EIMC_EVENT_QUAL) for EIM connection events indicate whether or not the attempt to connect to an EIM domain succeeded or failed due to security events (for example a bad password was specified). The possible values are:

Event qualifier (EIMC_EVENT_QUAL field in the tabular record)	Event qualifier value in the SMF Type 83 subtype 2 record	Event description
SUCCESS	00	Successful connect to the domain controller or a disconnection from the domain controller
BINDFAIL	03	Unauthorized to connect to the domain controller

*Table 44. Event-specific fields for EIM lookup events (EIML_EVENT_TYPE is "*LOOKUP")*

Field Name	Type	Length	Position		Comments
			Start	End	
EIML_API	Char	128	3000	3127	EIM API Name
EIML_DOMAIN_URL	Char	128	3130	3257	Domain URL
EIML_CONNECT_TYPE	Char	32	3260	3291	Connection type - SIMPLE, SIMPLE AND CRAM-MD5, SIMPLE CRAM_MD5 OPTIONAL, KERBEROS, SSL CLIENT AUTH
EIML_BIND_USER	Char	512	3294	3805	Bind user
EIML_REG_SRC	Char	64	3808	3871	Registry name - source
EIML_REG_USER_SRC	Char	64	3874	3937	Registry user name - source
EIML_IDENT_UUID	Char	36	3940	3975	Identifier UUID (fixed length)
EIML_IDENT_UNIQUE	Char	256	3978	4233	Identifier unique name
EIML_IDENT_ALIAS	Char	512	4236	4747	Identifier alias
EIML_REG_TGT	Char	64	4750	4813	Registry name - Target
EIML_REG_USER_INFO	Char	128	4816	4943	Registry user additional info
EIML_REG_USER_TGT	Char	64	4946	5009	Registry user name - Target

*Table 44. Event-specific fields for EIM lookup events (EIML_EVENT_TYPE is "*LOOKUP") (continued)*

Field Name	Type	Length	Position		Comments
			Start	End	
EIML_ASSOCPOL_TYPE	Char	32	5012	5043	Association or Policy Type - EIM_ADMIN, EIM_ALL_ASSOC, EIM_ALL_POLICY_ASSOC, EIM_CERT_FILTER_POLICY, EIM_DEFAULT_DOMAIN_POLICY, EIM_DEFAULT_REG_POLICY, EIM_SOURCE, EIM_SOURCE_AND_TARGET, EIM_TARGET
EIML_POL_FILT	Char	512	5046	5557	Policy Filter Value
EIML_REG_NAME	Char	64	5560	5623	Registry name or Registry name - ADMIN
EIML_REG_USER_NAME	Char	64	5626	5689	Registry user name

The event qualifiers (EIML_EVENT_QUAL) for EIM lookup events indicate whether or not the attempt to retrieve a user ID mapping was found, not found, or failed due to security reasons (the bind distinguished name does not have authority to retrieve the user ID mapping). The possible EIM lookup qualifier values are:

Event qualifier (EIML_EVENT_QUAL field in the tabular record)	Event qualifier value in the SMF Type 83 subtype 2 record	Event description
SUCCESS	00	Successful request
INSAUTH	01	Insufficient authority to retrieve EIM data
NOTFOUND	02	Mapping not found, the user was not authorized to access the EIM data, or policies or lookups were not enabled

*Table 45. Event-specific fields for EIM administrative events requiring changes to an EIM domain, registry, or user access (EIMD_EVENT_TYPE is "*ADMIN1")*

Field Name	Type	Length	Position		Comments
			Start	End	
EIMD_API	Char	128	3000	3127	EIM API Name
EIMD_DOMAIN_URL	Char	128	3130	3257	Domain URL
EIMD_CONNECT_TYPE	Char	32	3260	3291	Connection type - SIMPLE, SIMPLE AND CRAM-MD5, SIMPLE CRAM_MD5 OPTIONAL, KERBEROS, SSL CLIENT AUTH
EIMD_BIND_USER	Char	512	3294	3805	Bind user
EIMD_CERT_LABEL	Char	128	3808	3935	Certificate
EIMD_KEY_RING	Char	128	3938	4065	Key Ring
EIMD_ACCESS_TYPE	Char	32	4068	4099	Access type - EIM_ACCESS_ADMIN, EIM_ACCESS_REG_ADMIN, EIM_ACCESS_REGISTRY, EIM_ACCESS_IDENTIFIER_ADMIN, EIM_ACCESS_MAPPING_LOOKUP
EIMD_ACCESS_USER	Char	128	4102	4229	Access user

*Table 45. Event-specific fields for EIM administrative events requiring changes to an EIM domain, registry, or user access (EIMD_EVENT_TYPE is "*ADMIN1") (continued)*

Field Name	Type	Length	Position		Comments
			Start	End	
EIMD_ASSOCPOL_TYPE	Char	32	4232	4263	Association or Policy Type - EIM_ADMIN, EIM_ALL_ASSOC, EIM_ALL_POLICY_ASSOC, EIM_CERT_FILTER_POLICY, EIM_DEFAULT_DOMAIN_POLICY, EIM_DEFAULT_REG_POLICY, EIM_SOURCE, EIM_SOURCE_AND_TARGET, EIM_TARGET
EIMD_CHANGE_TYPE	Char	32	4266	4297	Change Type - EIM_ADD, EIM_CHG, EIM_ENABLE, EIM_DISABLE, EIM_RMV
EIMD_RESERVED_03	Char	32	4300	4331	Reserved
EIMD_DOMAIN_DESC	Char	64	4334	4397	Domain Description
EIMD_OPTIONS	Char	128	4400	4527	Options (multi-valued) - EIM_FAIL, EIM_GEN_UNIQUE, EIM_REGISTRY_MAPPING_LOOKUP, EIM_REGISTRY_POLICY_ASSOCIATIONS
EIMD_REG_ALIASTYPE	Char	32	4530	4561	Registry alias type - RACF, OS400, KERBEROS, AIX, NDS, LDAP, PD, WIN2K, others
EIMD_REG_ALIAS	Char	128	4564	4691	Registry alias value
EIMD_REG_DESC	Char	64	4694	4757	Registry description
EIMD_REG_NAME	Char	64	4760	4823	Registry name or Registry name - ADMIN
EIMD_RESERVED_04	Char	62	4826	4887	Reserved
EIMD_REG_SYSTEM	Char	64	4890	4953	Registry name - System
EIMD_REG_TYPE	Char	32	4956	4987	OID for the Registry type - RACF, OS400, KERBEROS-EX, KERBEROS-IG, AIX, NDS, LDAP, POLICY DIRECTOR, WIN2K, oid-CASEIGNORE, oid-CASEEXACT, others. See definitions in eim.h
EIMD_REG_URI	Char	128	4990	5117	Registry URI

The event qualifiers (EIMD_EVENT_QUAL) for EIM administrative events involving an EIM domain, registry, or user access indicate whether or not the change was successful or failed for security reasons (the bind distinguished name does not have authority to connect to the EIM domain or to change the EIM domain, registry, or user access as requested). The possible event qualifier values for these administrative events are:

Event qualifier (EIMD_EVENT_QUAL field in the tabular record)	Event qualifier value in the SMF Type 83 subtype 2 record	Event description
SUCCESS	00	Successful request
INSAUTH	01	Insufficient authority to modify the EIM domain or retrieve information from the domain
BINDFAIL	03	Unauthorized to connect to the domain controller

*Table 46. Event-specific fields for EIM administrative events involving changes to the identifiers, associations, and policies in an EIM domain (EIMI_EVENT_TYPE is "*ADMIN2")*

Field Name	Type	Length	Position		Comments
			Start	End	
EIMI_API	Char	128	3000	3127	EIM API Name
EIMI_DOMAIN_URL	Char	128	3130	3257	Domain URL
EIMI_CONNECT_TYPE	Char	32	3260	3291	Connection type - SIMPLE, SIMPLE AND CRAM-MD5, SIMPLE CRAM_MD5 OPTIONAL, KERBEROS, SSL CLIENT AUTH
EIMI_BIND_USER	Char	512	3294	3805	Bind user
EIMI_REG_SRC	Char	64	3808	3871	Registry name - source
EIMI_REG_USER_SRC	Char	64	3874	3937	Registry user name - source
EIMI_IDENT_UUID	Char	36	3940	3975	Identifier UUID (fixed length)
EIMI_IDENT_UNIQUE	Char	256	3978	4233	Identifier unique name
EIMI_IDENT_ALIAS	Char	512	4236	4747	Identifier alias
EIMI_REG_TGT	Char	64	4750	4813	Registry name - Target
EIMI_REG_USER_INFO	Char	128	4816	4943	Registry user additional info
EIMI_REG_USER_TGT	Char	64	4946	5009	Registry user name - Target
EIMI_IDENT_ALIASAT	Char	256	5012	5267	Identifier alias attribute value
EIMI_RESERVED_03	Char	290	5270	5559	Reserved
EIMI_ASSOCPOL_TYPE	Char	32	5562	5593	Association or Policy Type - EIM_ADMIN, EIM_ALL_ASSOC, EIM_ALL_POLICY_ASSOC, EIM_CERT_FILTER_POLICY, EIM_DEFAULT_DOMAIN_POLICY, EIM_DEFAULT_REG_POLICY, EIM_SOURCE, EIM_SOURCE_AND_TARGET, EIM_TARGET
EIMI_CHANGE_TYPE	Char	32	5596	5627	Change Type - EIM_ADD, EIM_CHG, EIM_ENABLE, EIM_DISABLE, EIM_RMV
EIMI_IDENT_INFO	Char	128	5630	5757	Identifier additional information
EIMI_IDENT_DESC	Char	64	5760	5823	Identifier description
EIMI_OPTIONS	Char	128	5826	5953	Options (multi-valued) - EIM_FAIL, EIM_GEN_UNIQUE, EIM_REGISTRY_MAPPING_LOOKUP, EIM_REGISTRY_POLICY_ASSOCIATIONS
EIMI_POL_FILT_TYPE	Char	32	5956	5987	Policy Filter Type - EIM_ALL_FILTERS, EIM_CERTIFICATE_FILTER
EIMI_POL_FILT	Char	512	5990	6501	Policy Filter Value
EIMI_REG_NAME	Char	64	6504	6567	Registry name or Registry name - ADMIN
EIMI_RESERVED_04	Char	62	6570	6631	Reserved
EIMI_REG_USER_DESC	Char	64	6634	6697	Registry user description
EIMI_REG_USER_NAME	Char	64	6700	6763	Registry user name

The event qualifiers (EIMI_EVENT_QUAL) for EIM administrative events involving identifiers, associations, and policies indicate whether or not the change was

successful or failed for security reasons (the bind distinguished name does not have the authority to make the requested change). The possible event qualifier values for these administrative events are:

Event qualifier (EIMI_EVENT_QUAL field in the tabular record)	Event qualifier value in the SMF Type 83 subtype 2 record	Event description
SUCCESS	00	Successful request
INSAUTH	01	Insufficient authority to modify the EIM domain or retrieve information from the domain

Included in z/OS are sample DB2 statements that can be used to load the EIM tabular output into DB2 for further analysis.

The sample DB2 statements for defining the table space and tables used to contain the EIM data is in SYS1.SAMPLIB(ITYSDBTB).

The sample DB2 load utility statements are in SYS1.SAMPLIB(ITYSDBLD). They can be used to load the EIM tabular output records of SMF unload into tables created by ITYSDBTB.

Chapter 11. EIM APIs

Programming Interface information

The EIM APIs are a programming Interface. They are intended for customers to use in customer-written programs.

End of Programming Interface information

This chapter provides information about EIM APIs, which are in alphabetical order. Before the APIs themselves, preliminary sections identify the authority to use the APIs and describe the EIM return code parameter, EimRC.

This chapter discusses C/C++ as well as Java APIs.

Authority to use APIs

To use most of the APIs, you must meet one of the following:

- Be an LDAP administrator
- Belong to an EIM-defined LDAP access control group

Different access groups can update or view different portions of the EIM domain and, therefore, have the authority to use different APIs.

Some APIs require additional authority granted through RACF profiles.

For information on access authorities, refer to “EIM access control” on page 30. The following APIs do not require the user to have an EIM authority:

- eimSetConfiguration
- eimRetrieveConfiguration
- eimCreateHandle
- eimDestroyHandle
- eimSetAttribute
- eimGetAttribute
- eimConnect
- eimConnectToMaster

Java APIs

Java APIs make EIM available to z/OS applications and servers written in Java. If your location uses programs written in Java and you wish to incorporate these EIM into these programs, you'll be able to use the Java EIM APIs to interface with EIM. Some benefits of this include:

- The ability to write EIM administrative and lookup applications in Java
- z/OS qualities of service in the Java-enabled APIs, specifically:
  1. Auditors can begin to track work requests as they are processed by their @server systems. Activity by the user IDs on different platforms may be traced back to the enterprise identifier
  2. Applications can use RACF profiles to store bind credentials (bind DNs and passwords) instead of less secure practices such as hard coding the values in their programs, defining side files to store the information, or creating yet another application unique mechanism for managing bind credentials

3. Applications can use RACF profiles to store the registry names instead of hard coding them in the programs, or store them in configuration files with application unique tools for managing the information. RACF allows the stored registry names to be shared among all applications on the system.

- Support Kerberos and CRAM-MD5 authentication mechanisms through the JNI provider
- Use existing system identities and the new distributed technologies interchangeably
- Administrators will have a central repository for saving relationships between enterprise identifiers and user IDs. As EIM based administration applications become available and operating systems exploit EIM, creation and deletion of user IDs can be tied to identifiers in an EIM domain
- The combination of EIM and LDAP eliminates the need for creating new user identities and the software to manage them. EIM and LDAP can replace the need for costly infrastructure to distribute identities to different platforms
- The ability to transform identities to local representations across operating systems and use existing system authorization mechanisms to validate access to system resources.

Authorization to use EIM Services

Authorization details are included in the documentation for the Java APIs, including those methods that have special authorization requirements. This documentation is separate from the z/OS library. For more details see “Obtaining documentation for the Java APIs” on page 163.

Mapping C++ to Java APIs

The following table shows the Java equivalent to the EIM C/C++ API. Several calls to java methods may be required to obtain the same information as a single C/C++ API. In addition, some of the Java methods are used in several places. Use the table to help guide you with answers to any questions you may have about RACF authorization requirements or what information will be audited for the Java APIs.

Table 47. C++ to Java API mapping

	EIM Java Interface Classes	
C/C++ API	**Class**	**Method(s)**
eimAddAccess	com.ibm.eim.AccessContext	addAdminAccessUser, addRegistryAccessUser
eimAddApplicationRegistry	com.ibm.eim.Domain	addApplicationRegistry
eimAddAssociation	com.ibm.eim.Eid	addAssociation
eimAddIdentifier	com.ibm.eim.Domain	addEid, addUniqueEid
eimAddPolicyAssociation	com.ibm.eim.Domain	addCertificateFilterPolicyAssociation, addDefaultDomainPolicyAssociation, addDefaultRegistryPolicyAssociation
eimAddPolicyFilter	com.ibm.eim.Registry	addCertificateFilter
eimAddSystemRegistry	com.ibm.eim.Domain	addSystemRegistry
eimChangeDomain	com.ibm.eim.Domain	setDescription
eimChangeIdentifier	com.ibm.eim.Eid	addAdditionalInfo, addAlias, removeAdditionalInfo, removeAlias, setDescription

Table 47. C++ to Java API mapping (continued)

C/C++ API	EIM Java Interface Classes	
	Class	**Method(s)**
eimChangeRegistry	com.ibm.eim.Registry	setDescription
	com.ibm.eim.SystemRegistry	setUri
eimChangeRegistryAlias	com.ibm.eim.Registry	addAlias, removeAlias
eimChangeRegistryUser	com.ibm.eim.RegistryUser	addAdditionalInfo, removeAdditionalInfo, setDescription
eimConnect	com.ibm.eim.DomainManager	getDomain
eimConnectToMaster	n/a	
eimCreateDomain	com.ibm.eim.DomainManager	createDomain
eimCreateHandle	n/a	
eimDeleteDomain	com.ibm.eim.Domain	delete
eimDestroyHandle	com.ibm.eim.Domain	disconnect
eimErr2String	com.ibm.eim.jni.EimException	
eimFormatPolicyFilter	com.ibm.eim.Formatter	formatCertificateFilter
eimFormatUserIdentity	com.ibm.eim.Formatter	formatRegistryUserName
eimGetAssociatedIdentifi ers	com.ibm.eim.Eid	getAssociations
	com.ibm.eim.RegistryUser	getAssociatedEids
eimGetAttribute	com.ibm.eim.Domain	getDn, getHost, getName, getPort, getURL, isSSL
eimGetRegistryNameFrom Alias	com.ibm.eim.Domain	getRegistriesByAlias, getRegistryNames
eimGetTargetFromIdentifier	com.ibm.eim.Eid	findTarget
eimGetTargetFromSource	com.ibm.eim.Domain	findTargetFromSource
eimGetVersion	n/a	
eimListAccess	com.ibm.eim.AccessContext	getAdminAccessUsers, getRegistryAccessUsers
eimListAssociations	com.ibm.eim.Association	getAssociationType, getEid, getRegistry, getRegistryName, getUid (a registry user identity)
	com.ibm.eim.Eid	getAssociations
eimListDomains	com.ibm.eim.Domain	getDescription
	com.ibm.eim.DomainManager	getDomains
eimListIdentifiers	com.ibm.eim.Domain	getEids, getEidsByAlias, getEidsByName, getEidsByUuid
	com.ibm.eim.Eid	getAdditionalInfo, getAliases, getDescription, getName, getUuid
eimListPolicyFilters	com.ibm.eim.Association	getPolicyFilter, getSourceRegistry, getSourceRegistryName
	com.ibm.eim.PolicyFilter	getFilterValue getPolicyFilterType, getSourceRegistry, getSourceRegistryName

Table 47. C++ to Java API mapping (continued)

C/C++ API	EIM Java Interface Classes	
	Class	**Method(s)**
eimListRegistries	com.ibm.eim.Domain	getRegistries
	com.ibm.eim.ApplicationRegistry	getSystemRegistryName
	com.ibm.eim.Registry	getDescription, getKind, getName, getType, getUUID
	com.ibm.eim.SystemRegistry	getUri
eimListRegistryAliases	com.ibm.eim.Domain	getRegistryNames
	com.ibm.eim.Registry	getAliases
eimListRegistry Associations	com.ibm.eim.Domain	getRegistryAssociations
eimListRegistryUsers	com.ibm.eim.Registry	getUsers
	com.ibm.eim.RegistryUser	getAdditionalInfo, getDescription
eimListUserAccess	com.ibm.eim.AccessContext	getUserAccess
eimQueryAccess	com.ibm.eim.AccessContext	getUserAccess, queryAdminUserAccess, queryRegistryUserAccess
	com.ibm.eim.UserAccess	hasAdmin, hasEidAdmin, hasMappingLookup, hasRegistryAdmin
eimRemoveAccess	com.ibm.eim.AccessContext	deleteAdminAccessUser, deleteRegistryAccessUser
eimRemoveAssociation	com.ibm.eim.Eid	removeAssociation
eimRemoveIdentifier	com.ibm.eim.Eid	delete
eimRemovePolicy Association	com.ibm.eim.Domain	removeCertificateFilterPolicyAssociation, removeDefaultDomainPolicyAssociation, removeDefaultRegistryPolicyAssociation
eimRemovePolicyFilter	com.ibm.eim.Registry	removeCertificateFilter
eimRemoveRegistry	com.ibm.eim.Registry	delete
eimRetrieveConfiguration	com.ibm.eim.jni.EimConfiguration Mgr	retrieveConfiguration (not supported on z/OS), retrieveConfiguration_z
eimSetConfiguration (not supported on z/OS)	com.ibm.eim.jni.EimConfiguration Mgr	setConfiguration (not supported on z/OS)
eimSetConfigurationExt	com.ibm.eim.jni.EimConfiguration Mgr	setConfiguration_z, removeConfiguration_z

Table 47. C++ to Java API mapping (continued)

C/C++ API	EIM Java Interface Classes	
	Class	**Method(s)**
Other public methods	com.ibm.eim.Domain	getAccessContext, getPolicyAssociationStatus, isConnected, setPolicyAssociationStatus
	com.ibm.eim.ConnectInfo	ConnectInfo, getEnvironment, getSSLInfo, isSSL, setSSL
	com.ibm.eim.jni.EimConfig	getKerberosRegistry, getLdapURL, getLocalRegistry, getProfileBindDn, getProfileBindPw, getProfileClassName, getProfileName, getX509Registry, isEnabled
	com.ibm.eim.DomainManager	getInstance
	com.ibm.eim.Formatter	getInstance
	com.ibm.eim.jni.EimException	EimException, getMessage, getRootException, getReasonCode, getSubstitutions, setRootException, setReasonCode, setSubstitutions, toString
	com.ibm.eim.Registry	getMappingLookupStatus, getPolicyAssociationStatus, getPolicyFilters, setMappingLookupStatus, setPolicyAssociationStatus
	com.ibm.eim.RegistryAlias	equals, getName, getType, hashCode
	com.ibm.eim.RegistryUser	getRegistryName, getTargetUserName
	com.ibm.eim.SSLInfo	SSLInfo, getCertificateLabel, getTrustStore, getTrustStorePw, getKeyStore, getKeyStorePw
	com.ibm.eim.UserAccess	getDN, getRegistries

Obtaining documentation for the Java APIs

The classes and methods for the EIM Java APIs are described in html files known as Javadoc. The Javadoc is the primary source of documentation for the application programmer using the EIM Java APIs.

Javadoc is generated as html using the Javadoc tool. Since a browser is needed to view the Javadoc, the Javadoc must either be downloaded to the user's workstation or viewed directly from an IBM public web site.

The Javadoc for the EIM Java APIs is available in eimzOS_DOC.jar. Javadoc is in html format and in ASCII encoding. The jar file can be downloaded to the workstation using ftp or other file transfer tool and then unjared for viewing locally.

To access the Javadoc files, do the following:

1. Extract the files from the eimzOS_DOC.jar using the jar utility
2. Open the file overview-summary.html using your browser. This file gives an overview of the APIs and contains hyperlinks to descriptions of the interfaces, classes, and methods.
3. Follow the HELP hyperlink or open the file help-doc.html to a general description of how Javadoc is organized.

EimRC -- EIM return code parameter for C/C++

All EIM APIs return an errno. If the EimRC parameter is not NULL, this EIM return code structure contains additional information about the error that was returned. You can use this to get a text description of the error.

The layout for EimRC follows:

```
typedef struct EimRC {
     unsigned int memoryProvidedByCaller;     /* Input: Size of the entire RC
                                                 structure. This is filled in by
                                                 the caller. This is used to tell
                                                 the API how much space was provided
                                                 for substitution text               */
     unsigned int memoryRequiredToReturnData; /* Output: Filled in by API
                                                 to tell caller how much data could
                                                 have been returned. Caller can then
                                                 determine if the caller provided
                                                 enough space (i.e. if the entire
                                                 substitution string was able to be
                                                 copied to this structure).          */
     int returnCode;                          /* Same as the errno returned as the
                                                 rc for the API                      */
     int messageCatalogSetNbr;                /* Message catalog set number          */
     int messageCatalogMessageID;             /* Message catalog message id          */
     int ldapError;                           /* ldap error, if available            */
     int sslError;                            /* ssl error, if available             */
     char reserved[16];                       /* Reserved for future use             */
     unsigned int substitutionTextLength;     /* Length of substitution text
                                                 excluding a null-terminator which
                                                 may or may not be present           */
     char substitutionText[1];                /* Further info describing the
                                                   error.                            */
} EimRC;
```

Field descriptions

memoryProvidedByCaller
(Input)

The number of bytes the calling application provides for the error code. The number of bytes provided must be at least 48.

memoryRequiredToReturnData
(Output)

The length of the error informational message, in bytes, that is necessary for the API to return. If this is 0, no error was detected and none of the fields that follow this field in the structure are changed.

returnCode
(Output)

The errno returned for this API. This is the same as the return value for each API.

messageCatalogSetNbr
(Output)

The message set number for the EIM catalog. You can use this with the messageCatalogID to get the error message text.

messageCatalogMessageID
(Output)

The message ID number for the EIM catalog. You can use this with the messageCatalogSetNbr to get the error message text.

ldapError
(Output)

An error code returned by an LDAP client API. The interpretation of the error code is in the substitution text.

sslError
(Output)

An error code returned by an LDAP client API. If not zero, this value will be displayed in the substitution text as the SSL reason code. Refer to the ldapssl.h header file in the LDAP Client API document for further information.

reserved
(Output)

Reserved for future use.

substitutionTextLength
(Output)

This field is set if any substitution text is returned. If there is no substitution text, this field is zero.

substitutionText
(Output)

Message substitution text.

eimAddAccess

Purpose

Adds the user to an EIM access group identified by the access type.

Format

```
#include <eim.h>


int eimAddAccess(EimHandle          * eim,
                 EimAccessUser      * accessUser,
                 enum EimAccessType   accessType,
                 char               * registryName,
                 EimRC              * eimrc)
```

Parameters

eim

(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

accessUser

(Input) A structure that contains the name of the user requiring access. The user name can be:

- A distinguished name
- A Kerberos Principal

EIM_ACCESS_DN Indicates a distinguished name defined in an LDAP directory that you can use to bind to the EIM domain.

EIM_ACCESS_LOCAL_USER (z/OS does not support this. Use EIM_ACCESS_DN instead.) It indicates a local user name on the system where the API runs. The local user name is converted to the appropriate access ID for this system.

EIM_ACCESS_KERBEROS Indicates a Kerberos identity, which is converted to the appropriate access ID. For example, `petejones@therealm` is converted to `ibm-kn=petejones@threalm`.

The EimAccessUser structure layout follows:

```
enum EimAccessUserType {
        EIM_ACCESS_DN,
        EIM_ACCESS_KERBEROS,
        EIM_ACCESS_LOCAL_USER
  };
typedef struct EimAccessUser
{
    union {
        char *dn;
        char *kerberosPrincipal;
        char *localUser;
      }user;
      enum EimAccessUserType userType;
}EimAccessUser;
```

accessType
(Input) The type of access to add. Valid values are:

EIM_ACCESS_ADMIN (0) Administrative authority to the entire EIM domain.

EIM_ACCESS_REG_ADMIN (1)
Administrative authority to all registries in the EIM domain.

EIM_ACCESS_REGISTRY (2) Administrative authority to the registry specified in the registryName parameter.

EIM_ACCESS_IDENTIFIER_ADMIN (3)
Administrative authority to all of the identifiers in the EIM domain.

EIM_ACCESS_MAPPING_LOOKUP (4)
Authority to perform mapping lookup operations.

registryName
(Input) The name of the registry for which to add access. Registry names are case-independent (meaning not case-sensitive). This parameter is used only if accessType is EIM_ACCESS_REGISTRY. If accessType is anything other than EIM_ACCESS_REGISTRY, this parameter must be NULL.

The following special characters are not allowed in registry names:

```
, = + < > # ; \ *
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimListAccess” on page 286
- “eimListUserAccess” on page 344
- “eimQueryAccess” on page 351
- “eimRemoveAccess” on page 355

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.)Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_ACCESS_TYPE_INVAL (2)	Access type is not valid.
	EIMERR_ACCESS_USERTYPE_INVAL (3)	Access user type is not valid.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_REG_MUST_BE_NULL (55)	Registry name must be NULL when access type is not EIM_ACCESS_REGISTRY.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete the operation. Use the eimConnectToMaster API to get a write connection.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error. %s
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following illustrates adding the distinguished name of a user to the EIM Administrator access group.

```
#include <eim.h>

.
 .
```

```
.
   int           rc;
   char          eimerr[200];
   EimRC       * err;
   EimHandle     handle;
   EimAccessUser user;
.
.
.
   /* Set up error structure.                */
   memset(eimerr,0x00,200);
   err = (EimRC *)eimerr;
   err->memoryProvidedByCaller = 200;
.
.
.
   /* Set up access user information         */
   user.userType = EIM_ACCESS_DN;
   user.user.dn="cn=pete,o=ibm,c=us";

   /* Add access for this user.              */
   rc = eimAddAccess(&handle,
                     &user,
                     EIM_ACCESS_ADMIN,
                     NULL,
                     err);
.
.
.
```

eimAddApplicationRegistry

Purpose

Adds an application registry to the EIM domain. An application registry contains a subset of a system registry's user IDs.

Format

```
#include <eim.h>


int eimAddApplicationRegistry(EimHandle      * eim,
                              char           * registryName,
                              char           * registryType,
                              char           * description,
                              char           * systemRegistryName,
                              EimRC          * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

registryName
(Input) The name for this application registry. This name cannot have a NULL value and must be unique within the EIM domain. The uniqueness of the registry name is for the domain and not for the system registry that the application registry belongs to. Registry names are case-independent (meaning, not case-sensitive).

The following special characters are not allowed in registry names.

```
, = + < > # ; \ *
```

registryType
(Input) A string form of an OID that represents the registry type and a user name normalization method. The normalization method is necessary because some registries are case-independent while others are case-dependent. EIM uses this information to make sure the appropriate search occurs. When a registry is case-independent, registry user names are converted to uppercase. See "eim.h" on page 395 for more information. Users can define their own registry types. See "EIM registry definition" on page 13 for details.

The following are possible registry types:

- EIM_REGTYPE_RACF
- EIM_REGTYPE_OS400
- EIM_REGTYPE_KERBEROS_EX
- EIM_REGTYPE_KERBEROS_IG
- EIM_REGTYPE_WIN_DOMAIN_KERB_IG
- EIM_REGTYPE_AIX
- EIM_REGTYPE_NDS
- EIM_REGTYPE_LDAP
- EIM_REGTYPE_POLICY_DIRECTOR
- EIM_REGTYPE_TIVOLI_ACCESS_MANAGER
- EIM_REGTYPE_WIN2K
- EIM_REGTYPE_WINDOWS_LOCAL_WS

- EIM_REGTYPE_X509
- EIM_REGTYPE_LINUX
- EIM_REGTYPE_DOMINO_LONG
- EIM_REGTYPE_DOMINO_SHORT

description
(Input) The description for this new application registry. This parameter can be NULL.

systemRegistryName
(Input) The name of the system registry of which this application registry is a subset. This parameter cannot be NULL.

The following special characters are not allowed in registry names.

```
, = + < > # ; \ *
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddSystemRegistry” on page 190
- “eimChangeRegistry” on page 203
- “eimListRegistries” on page 317
- “eimRemoveRegistry” on page 374

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.

Return Value	Meaning	
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EEXIST	EIM registry entry already exists.	
	EIMERR_REGISTRY_EXISTS (37)	The registry entry already exists within the particular domain.
EINVAL	Input parameter was not valid.	
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_CHAR_INVAL (21)	A restricted character was used in the object name. Check the API documentation for a list of restricted characters.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
ENOENT	System registry not found.	
	EIMERR_NO_SYSREG (33)	System registry not found.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete the operation. Use the eimConnectToMaster API to get a write connection.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates creating a new EIM application registry:

```
#include <eim.h>


.
 .
 .
    int           rc;
    char          eimerr[200];
    EimRC       * err;
    EimHandle     handle;

    /* Set up error structure.                */
    memset(eimerr,0x00,200);
```

```
   err = (EimRC *)eimerr;
   err->memoryProvidedByCaller = 200;

   /* Add new application registry            */
   rc = eimAddApplicationRegistry(&handle,
                                  "MyAppRegistry",
                                  EIM_REGTYPE_OS400,
                                  "For App applications",
                                  "MyRegistry",
                                  err);
.
.
.
```

eimAddAssociation

Purpose

Associates a local identity in a specified user registry with an EIM identifier. EIM supports three kinds of associations:

- Source
- Target
- Administrative

(See page “EIM associations” on page 17 for more information about these kinds of associations.)

All EIM associations are between an EIM identifier and a local user identity. An association is never directly between local user identities. For an EIM identifier to be useful in mapping lookup operations, it must have at least one ″source″ or at least one ″target″ association.

Associated source identities are user identities that are primarily for authentication purposes. You can use an associated source identity as the source identity of a mapping lookup operation (that is, with eimGetTargetFromSource), but you cannot find an associated source identity by making it the target of a mapping lookup operation.

Associated target identities are user identities that are primarily used to secure existing data. You can find an associated target identity as the result of a mapping lookup operation, but you cannot use an associated target identity as the source identity for a mapping lookup operation.

Administrative associations are used to show that an identity is associated with an EIM identifier. You cannot use an administrative association as the source or target of a mapping lookup operation.

You can use a single user identity as both a target and a source. You do this by creating both a source and a target association for the local user identity with the appropriate EIM identifier. Although this API supports an association type of EIM_SOURCE_AND_TARGET, it actually creates two associations.

Format

```
#include <eim.h>


int eimAddAssociation(EimHandle              * eim,
                      enum EimAssociationType  associationType,
                      EimIdentifierInfo      * idName,
                      char                   * registryName,
                      char                   * registryUserName,
                      EimRC                  * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

associationType
(Input) The type of association to add. Valid values are:

EIM_TARGET (1)
Add a target association.

EIM_SOURCE (2)
Add a source association.

EIM_SOURCE_AND_TARGET (3)
Add both a source association and a target association.

EIM_ADMIN (4)
Add an administrative association.

idName
(Input) A structure that contains the identifier name for this association. The layout of the EimIdentifierInfo structure follows:

```
enum EimIdType {
       EIM_UNIQUE_NAME,
       EIM_ENTRY_UUID,
       EIM_NAME
   };


   typedef struct EimIdentifierInfo
   {
       union {
           char       * uniqueName;
           char       * entryUUID;
           char       * name;
       } id;
       enum EimIdType        idtype;
   } EimIdentifierInfo;
```

`idtype`
The `idtype` in the EimIdentifierInfo structure indicates which identifier name has been provided. EIM_UNIQUE_NAME finds at most one matching identifier. EIM_NAME results in an error if your EIM domain has more than one identifier containing the same name.

registryName
(Input) The registry name for the association. Registry names are case-independent (meaning, not case-sensitive).

The following special characters are not allowed in registry names:

```
, = + < > # ; \ *
```

registryUserName
(Input) The registry user name for the association. The API normalizes the registry user name according to the normalization method for the defined registry. The registry user name should begin with a non-blank character.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, *eimrc* is set with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimGetAssociatedIdentifiers” on page 254
- “eimListAssociations” on page 291
- “eimRemoveAssociation” on page 359

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The authority that the access group has to the EIM data depends on the type of association being added.

For administrative and source associations, the access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM identifiers administrator

For target associations, the access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM registry *X* administrator

z/OS authorization

No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBADNAME	Registry or identifier name is not valid or insufficient access to EIM data.	
	EIMERR_IDNAME_AMBIGUOUS (20)	More than one EIM identifier was found that matches the requested identifier name.
	EIMERR_NOIDENTIFIER (25)	EIM identifier not found or insufficient access to EIM data.
	EIMERR_NOREG (28)	EIM registry not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.

Return Value	Meaning
EINVAL	Input parameter was not valid.
	EIMERR_ASSOC_TYPE_INVAL (4) Association type is not valid.
	EIMERR_HANDLE_INVAL (17) EimHandle is not valid.
	EIMERR_IDNAME_TYPE_INVAL (52) The EimIdType value is not valid.
	EIMERR_PARM_REQ (34) Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35) (z/OS does not return this value.) Pointer parameter is not valid.
EMVSERR	An MVS environment or internal error has occurred.
	EIMERR_ZOS_DATA_CONVERSION (6013) Error occurred when converting data between code pages.
ENOMEM	Unable to allocate required space.
	EIMERR_NOMEM (27) No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.
	EIMERR_NOT_CONN (31) Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.
	EIMERR_READ_ONLY (36) This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete the operation. Use the eimConnectToMaster API to get a write connection.
EUNKNOWN	Unexpected exception.
	EIMERR_LDAP_ERR (23) Unexpected LDAP error.
	EIMERR_UNEXP_OBJ_VIOLATION (56) Unexpected object violation.
	EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Example

The following example illustrates adding an administrative, source, and target association for the same identifier:

```
#include <eim.h>


.
 .
 .
    int           rc;
    char          eimerr[200];
    EimRC       * err;
    EimHandle     handle;
    EimIdentifierInfo x;

    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;
```

```
.
.
.
   /* Set up identifier information            */
   x.idtype = EIM_UNIQUE_NAME;
   x.id.uniqueName = "mjones";

   /* Add an admin association                 */
   rc = eimAddAssociation(&handle,
                          EIM_ADMIN,
                          &x,
                          "MyRegistry",
                          "maryjones",
                          err);
.
.
.
   /* Add a source association                 */
   rc = eimAddAssociation(&handle,
                          EIM_SOURCE,
                          &x,
                          "kerberosRegistry",
                          "mjjones",
                          err);
.
.
.
   /* Add a target association                 */
   rc = eimAddAssociation(&handle,
                          EIM_TARGET,
                          &x,
                          "MyRegistry",
                          "marryjo",
                          err);
.
.
.
```

eimAddIdentifier

Purpose

Creates an identifier in EIM related to a specific person or entity within an enterprise. This identifier is used to manage information and identify relationships for a specific user or identity.

Format

```
#include <eim.h>


int eimAddIdentifier (EimHandle        * eim,
                      char             * name,
                      enum EimIdAction   nameInUseAction,
                      unsigned int     * sizeOfUniqueName,
                      char             * uniqueName,
                      char             * description,
                      EimRC            * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

name
(Input) A name to use for this identifier.

The following characters are special characters that are not allowed in the identifier *name*.

```
, = + < > # ; \ *
```

nameInUseAction
(Input) The name for the new identifier must be unique. This value indicates the action to take if the provided name is already in use. Possible values are:

EIM_FAIL (0)
Do not generate a unique name; return an error.

EIM_GEN_UNIQUE (1)
Generate a unique name.

sizeOfUniqueName
(Input/Output) The size of the field in which to return the unique name. EIM ignores this parameter if *nameInUseAction* is `EIM_FAIL`. At input this parameter is the size the caller provides. On output it contains the actual size returned. This value should be the size of the name parameter plus an additional 20 bytes.

uniqueName
(Output) The space in which to return the unique identifier for this new EIM identifier. EIM ignores this parameter if *nameInUseAction* is `EIM_FAIL`.

description
(Input) Description for the new EIM identifier. This parameter can be NULL.

eimrc
(Input/Output) The structure in which to return error code information. If the

return value is not 0, EIM sets *eimrc* with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimChangeIdentifier” on page 199
- “eimGetAssociatedIdentifiers” on page 254
- “eimListIdentifiers” on page 305
- “eimRemoveIdentifier” on page 364

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM identifiers administrator

z/OS authorization

No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EEXIST	Identifier already exists.	
	EIMERR_IDENTIFIER_EXISTS (19)	EIM Identifier already exists by this name.

Return Value	Meaning	
EINVAL	Input parameter was not valid.	
	EIMERR_CHAR_INVAL (21)	A restricted character was used in the object name. Check the API for a list of restricted characters.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_IDACTION_INVAL (18)	Name in use action is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_UNIQUE_SIZE (43)	Length of unique name is not valid.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete the operation. Use the eimConnectToMaster API to get a write connection.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates adding an EIM identifier:

```
#include <eim.h>


   int           rc;
   char          eimerr[200];
   EimRC       * err;
   EimHandle     handle;
   char          unique[30];
   unsigned int  sizeOfUnique = 30;

   /* Set up error structure.                 */
   memset(eimerr,0x00,200);
   err = (EimRC *)eimerr;
   err->memoryProvidedByCaller = 200;
.
.
.

   /* Add new identifier of Mary Smith        */
   rc = eimAddIdentifier(&handle,
                         "Mary Smith",
                         EIM_GEN_UNIQUE,
                         &sizeOfUnique,
                         unique,
```

```
                                    "The coolest person",
                                    err);
                .
                .
                .
```

eimAddPolicyAssociation

Purpose

Add a policy mapping to a domain. Policy associations specify the target association for a mapping lookup operation without having to define specific source associations for all users. The three types of policies allowed are certificate, default registry, and default domain.

Format

```
#include <eim.h>
int eimAddPolicyAssociation(EimHandle * eim,
                            EimPolicyAssociationInfo * policyAssoc,
                            EimRC * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

policyAssoc
(Input) The information about the policy association to be added. Valid types are:

EIM_CERT_FILTER_POLICY
Used to map user (or client) certificates with similar attributes to the same target identity in the target registry.

EIM_DEFAULT_REG_POLICY
Used to map any user in the specified source registry to the same target identity in the target registry.

EIM_DEFAULT_DOMAIN_POLICY
Used to map all users to the same target identity in the target registry.

The structure layout is as follows:

```
 enum EimAssociationType {
       EIM_ALL_ASSOC,                /* Not supported on this interface*/
       EIM_TARGET,                   /* Not supported on this interface*/
       EIM_SOURCE,                   /* Not supported on this interface*/
       EIM_SOURCE_AND_TARGET,        /* Not supported on this interface*/
       EIM_ADMIN,                    /* Not supported on this interface*/
       EIM_CERT_FILTER_POLICY,       /* Policy is a certificate
                                        filter policy association.     */
       EIM_DEFAULT_REG_POLICY,       /* Policy is a default
                                        registry policy association    */
       EIM_DEFAULT_DOMAIN_POLICY     /* Policy is a default policy for
                                        the domain.                    */
 };
   typedef struct EimCertificateFilterPolicyAssociation
   {
       char  * sourceRegistry;       /* The source registry to add the
                                        policy association to.         */
       char  * filterValue;          /* The filter value of the policy.*/
       char  * targetRegistry;       /* The name of the target registry
                                        that the filter value should
                                        map to.                        */
       char  * targetRegistryUserName; /* The name of the target registry
                                        user name that the filter value
                                        should map to.                 */
   } EimCertificateFilterPolicyAssociation;
   typedef struct EimDefaultRegistryPolicyAssociation
```

```
{
    char  * sourceRegistry;         /* The source registry to add the
                                       policy association to.         */
    char  * targetRegistry;         /* The name of the target registry
                                       that the policy should map to. */
    char  * targetRegistryUserName; /* The name of the target registry
                                       user name that the policy
                                       should map to.                 */
} EimDefaultRegistryPolicyAssociation;
typedef struct EimDefaultDomainPolicyAssociation
{
    char  * targetRegistry;         /* The name of the target registry
                                       that the policy should map to. */
    char  * targetRegistryUserName; /* The name of the target registry
                                       user name that the policy
                                       should map to.                 */
} EimDefaultDomainPolicyAssociation;
typedef struct EimPolicyAssociationInfo
{
    enum EimAssociationType type;
    union {
        EimCertificateFilterPolicyAssociation   certFilter;
        EimDefaultRegistryPolicyAssociation     defaultRegistry;
        EimDefaultDomainPolicyAssociation       defaultDomain;
    } policyAssociation;

} EimPolicyAssociationInfo;
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimRemovePolicyAssociation” on page 367
- “eimListRegistryAssociations” on page 330
- “eimFormatPolicyFilter” on page 243
- “eimAddPolicyFilter” on page 187
- “eimChangeDomain” on page 194
- “eimChangeRegistry” on page 203
- “eimGetTargetFromSource” on page 276
- “eimGetTargetFromIdentifier” on page 270

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM Registries Administrator
- EIM authority to an individual registry. This authority is needed to the target registry.

Return Values

Return Value	Meaning
0	Request was successful.
EACCES	Access denied. Not enough permissions to access data. **EIMERR_ACCESS (1)** Insufficient access to EIM data.
EBADDATA	eimrc is not valid.
EBADNAME	Registry name is not valid, insufficient access to EIM data, or policy filter value is not found. **EIMERR_NOREG (28)** EIM Registry not found or insufficient access to EIM data. **EIMERR_NOPOLICYFILTER (61)** Policy filter value not found for the specified EIM Registry.
EBUSY	Unable to allocate internal system object. **EIMERR_NOLOCK (26)** (z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error. **EIMERR_DATA_CONVERSION (13)** (z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid. **EIMERR_CHAR_INVAL (21)** A restricted character was used in the object name. Check the API for a list of restricted characters. **EIMERR_FUNCTION_NOT_SUPPORTED (70)** The specified or configured EIM Domain controller does not support this API. **EIMERR_HANDLE_INVAL (17)** EimHandle is not valid. **EIMERR_PARM_REQ (34)** Missing required parameter. Check the API documentation. **EIMERR_PTR_INVAL (35)** (z/OS does not return this value.) Pointer parameter is not valid.
ENOMEM	Unable to allocate required space. **EIMERR_NOMEM (27)** No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made. **EIMERR_NOT_CONN (31)** Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	The connection is for read-only. Need to connect to master. **EIMERR_READ_ONLY (36)** This connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete this operation. Use eimConnectToMaster to get a write connection.
EUNKNOWN	Unexpected exception. **EIMERR_LDAP_ERR (23)** Unexpected LDAP error. **EIMERR_UNKNOWN (44)** Unknown error or unknown system state. **EIMERR_UNEXP_OBJ_VIOLATION (56)** Unexpected object violation.

Example

The following example adds a default registry policy association.

```
#include <eim.h>
#include <string.h>
.
.
.
    int                       rc;
    char                      eimerr[250];
    EimRC                   * err;
    EimHandle                 handle;
    EimPolicyAssociationInfo assocInfo;

    /* Set up error structure.                  */
    memset(eimerr,0x00,250);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 250;
.
.
.
    /* Set up policy association information    */
    assocInfo.type = EIM_DEFAULT_REG_POLICY;
    assocInfo.policyAssociation.defaultRegistry.sourceRegistry = "MySourceRegistry";
    assocInfo.policyAssociation.defaultRegistry.targetRegistry = "localRegistry";
    assocInfo.policyAssociation.defaultRegistry.targetRegistryUserName = "mjjones";

    /* Add the policy association               */
    rc = eimAddPolicyAssociation(&handle, &assocInfo, err);
.
.
.
```

eimAddPolicyFilter

Purpose

Adds a policy filter to the domain. A policy association can then be added to the filter value using the eimAddPolicyAssociation API.

Format

```
#include <eim.h>
int eimAddPolicyFilter(EimHandle *           eim,
                       EimPolicyFilterInfo * filterInfo,
                       EimRC *               eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

filterinfo
(Info) The information about the policy filter to be added. Valid types are:

EIM_CERTIFICATE_FILTER (1)
Identifies that all certificates issued by the same Certificate Authority (CA) are mapped to the same target identity in the target registry. Or, all certificates from the same organization are mapped to the same target identity in the target registry.

The structure layout is as follows:

```
 enum EimPolicyFilterType {
       EIM_ALL_FILTERS,               /* All policy filters -- not
                                         supported for this interface.  */
       EIM_CERTIFICATE_FILTER         /* Policy filter is a certificate
                                         filter.                        */
   };
   typedef struct EimCertificatePolicyFilter
   {
       char  * sourceRegistry;        /* The source registry to add the
                                         policy filter to.              */
       char  * filterValue;           /* The policy filter value.       */
   } EimCertificatePolicyFilter;
   typedef struct EimPolicyFilterInfo
   {
       enum EimPolicyFilterType type;
       union {
           EimCertificatePolicyFilter   certFilter;
       } filter;
    } EimPolicyFilterInfo;
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimRemovePolicyFilter” on page 371
- “eimListPolicyFilters” on page 312
- “eimFormatPolicyFilter” on page 243

- "eimAddPolicyAssociation" on page 183
- "eimRemovePolicyAssociation" on page 367
- "eimListRegistryAssociations" on page 330

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

Return Values

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBADNAME	Registry name is not valid, insufficient access to EIM data, or policy filter value is not found.	
	EIMERR_NOREG (28) EIM Registry not found or insufficient access to EIM data.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_CHAR_INVAL (21)	A restricted character was used in the object name. Check the API for a list of restricted characters.
	EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified or configured EIM Domain controller does not support this API.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.

Return Value	Meaning	
EROFS	The connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	This connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete this operation. Use eimConnectToMaster to get a write connection.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.
	EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.

Example

```
#include <eim.h>
#include <string.h>
.
.
.
    int                 rc;
    char                eimerr[250];
    EimRC             * err;
    EimHandle           handle;
    EimPolicyFilterInfo filterInfo;

    /* Set up error structure.                  */
    memset(eimerr,0x00,250);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 250;
.
.
.
    /* Set up policy filter information         */
    filterInfo.type = EIM_CERTIFICATE_FILTER;
    filterInfo.filter.certFilter.sourceRegistry = "MySourceRegistry";
    filterInfo.filter.certFilter.filterValue =
      "<IDN>OU=Certified User,O=A Certificate Authority,L=Internet</IDN>";

    /* Add the policy filter                    */
    rc = eimAddPolicyFilter(&handle,
                            &filterInfo,
                            err);
.
.
.
```

eimAddSystemRegistry

Purpose

Adds a system registry to the EIM domain. After you add it, this registry participates in the EIM domain. You can make mapping associations only with identities in registries that are currently participating in the EIM domain.

Format

```
#include <eim.h>


int eimAddSystemRegistry(EimHandle      * eim,
                         char           * registryName,
                         char           * registryType,
                         char           * description,
                         char           * URI,
                         EimRC          * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

registryName
(Input) The name for this system registry. This name cannot have a NULL value and must be unique within the EIM domain. Registry names are case-independent (meaning, not case-sensitive).

The following special characters are not allowed in registry names.

```
, = + < > # ; \ *
```

registryType
(Input) A string form of an OID that represents the registry type and a user name normalization method. The normalization method is necessary because some registries are case-independent and others are case-dependent. EIM uses this information to make sure the appropriate search occurs. When a registry is case-independent, registry user names are converted to uppercase. The following are possible registry types:

- EIM_REGTYPE_RACF
- EIM_REGTYPE_OS400
- EIM_REGTYPE_KERBEROS_EX
- EIM_REGTYPE_KERBEROS_IG
- EIM_REGTYPE_WIN_DOMAIN_KERB_IG
- EIM_REGTYPE_AIX
- EIM_REGTYPE_NDS
- EIM_REGTYPE_LDAP
- EIM_REGTYPE_POLICY_DIRECTOR
- EIM_REGTYPE_TIVOLI_ACCESS_MANAGER
- EIM_REGTYPE_WIN2K
- EIM_REGTYPE_WINDOWS_LOCAL_WS
- EIM_REGTYPE_X509
- EIM_REGTYPE_LINUX

- EIM_REGTYPE_DOMINO_LONG
- EIM_REGTYPE_DOMINO_SHORT

description
(Input) The description for this new system registry entry. This parameter can be NULL.

URI
(Input) The LDAP URI (Universal Resource Identifier) for this registry, if available.

If the system registry is accessable through LDAP, then for documentation purposes you can set the URI as the URL for the system registry.

Example:

If the following three premises are true:

- The LDAP server is running on z/OS with the host name some.ldap.host
- The LDAP server is listening on port 389
- The LDAP server is configured with the RACF SDBM and has the suffix `cn=RACFA,o=ibm,c=us`

Then the URI could be set to the following:

```
ldap://some.ldap.host:389/profileType=User,cn=RACFA,o=ibm,c=us
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddApplicationRegistry” on page 170
- “eimChangeRegistry” on page 203
- “eimListRegistries” on page 317
- “eimRemoveRegistry” on page 374

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning
0	Request was successful.

Return Value	Meaning	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EEXIST	EIM registry entry already exists.	
	EIMERR_REGISTRY_EXISTS (37)	The registry entry already exists within the particular domain.
EINVAL	Input parameter was not valid.	
	EIMERR_CHAR_INVAL (21)	A restricted character was used in the object name. Check the API for a list of restricted characters.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	The connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	This connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete this operation. Use eimConnectToMaster to get a write connection.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates creating a new EIM system registry:

```
#include <eim.h>

.
.
.

   int          rc;
```

```
   char          eimerr[200];
   EimRC       * err;
   EimHandle     handle;

   /* Set up error structure.                */
   memset(eimerr,0x00,200);
   err = (EimRC *)eimerr;
   err->memoryProvidedByCaller = 200;
.
.
.

   /* Add new system registry                */
   rc = eimAddSystemRegistry(&handle,
                             "MyRegistry",
                             EIM_REGTYPE_OS400,
                             "The first registry",
                             NULL, /* No URI specified for this registry */
                             err);
.
.
.
```

eimChangeDomain

Purpose

Changes an attribute for the EIM domain entry.

Format

```
#include <eim.h>


int eimChangeDomain(char               * ldapURL,
                    EimConnectInfo       connectInfo,
                    enum EimDomainAttr   attrName,
                    char               * attrValue,
                    enum EimChangeType   changeType,
                    EimRC              * eimrc)
```

Parameters

ldapURL

(Input) A uniform resource locator (URL) that contains the EIM host information. This parameter is required. This URL has the following format:

```
ldap://host:port/dn
```

or

```
ldaps://host:port/dn
```

host:port
Name of the host on which the EIM domain controller is running. (The port number is optional. If not specified, the default LDAP or LDAPS port will be used.)

dn Distinguished name of the domain to change.

Examples:

```
ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us
ldaps://eimsystem:636/ibm-eimDomainName=myEimDomain,o=ibm,c=us
```

Note: In contrast with `ldap`, `ldaps` indicates that this host and port combination uses SSL and TLS.

connectInfo

(Input) Connect information. This parameter provides the information required to bind to LDAP. If the system is configured to connect to a secure port, EimSSLInfo is required.

For the EIM_SIMPLE connect type, the `creds` field should contain the EimSimpleConnectInfo structure with a binddn and password.

EimPasswordProtect determines the level of password protection on the LDAP bind.

EIM_PROTECT_NO (0) The clear-text password is sent on the bind.

EIM_PROTECT_CRAM_MD5 (1)
The protected password is sent on the bind. The server side must support cram-md5 protocol to send the protected password.

EIM_PROTECT_CRAM_MD5_OPTIONAL (2)
The protected password is sent on the bind if

the cram-md5 protocol is supported. Otherwise, the clear-text password is sent.

For EIM_KERBEROS, the default logon credentials are used. The `kerberos_creds` field must be NULL.

For EIM_CLIENT_AUTHENTICATION, the `creds` field is ignored. The `ssl` field must point to a valid EimSSLInfo structure. The `keyring` field is required in the EimSSLInfo structure. It can be the name of a System SSL key database file or a RACF keyring name. The `keyring_pw` field is required when the keyring is the name of a System SSL key database field. The `certificateLabel` field is optional. If it is NULL the default certificate in the keyring is used.

The structure layouts follow:

```
enum EimPasswordProtect {
    EIM_PROTECT_NO,
    EIM_PROTECT_CRAM_MD5,
    EIM_PROTECT_CRAM_MD5_OPTIONAL
};

enum EimConnectType {
    EIM_SIMPLE,
    EIM_KERBEROS,
    EIM_CLIENT_AUTHENTICATION
};


typedef struct EimSimpleConnectInfo
{
     enum EimPasswordProtect protect;
     char * bindDn;
     char * bindPw;
} EimSimpleConnectInfo;

typedef struct EimSSLInfo
{
     char * keyring;
     char * keyring_pw;
     char * certificateLabel;
} EimSSLInfo;


typedef struct EimConnectInfo
{
     enum EimConnectType type;
     union {
         gss_cred_id_t * kerberos;
         EimSimpleConnectInfo simpleCreds;
     } creds;
   EimSSLInfo * ssl;
} EimConnectInfo;
```

attrName

(Input) The attribute to be updated. Valid values are:

EIM_DOMAIN_DESCRIPTION (0)

Changes the description for the EIM domain. Valid `changeType` is EIM_CHG (0).

EIM_DOMAIN_POLICY_ASSOCIATIONS (1)

Change the indicator for whether or not the domain supports policy associations in a mapping lookup. By default, the policy associations are not enabled. Valid changeType is EIM_ENABLE (3) or EIM_DISABLE(4).

attrValue
(Input) The new value for the attribute. This value can be a NULL string (for example, ″″). If the attribute being changed is EIM_DOMAIN_POLICY_ASSOCIATIONS, this value must be NULL.

changeType
(Input) The type of change to make. This could be add, remove, or change, enable or disable. The *attrName* parameter indicates which type is allowed for each attribute. Valid values are:

EIM_CHG (0)
The attribute is set to the new value.

EIM_ENABLE (3)
Allow policy associations in a mapping lookup.

EIM_DISABLE (4)
Do not allow policy associations in a mapping lookup.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, *eimrc* is set with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimCreateDomain” on page 225
- “eimDeleteDomain” on page 234
- “eimListDomains” on page 298

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBADNAME	EIM domain not found or insufficient access to EIM data.	
	EIMERR_NODOMAIN (24)	EIM domain not found or insufficient access to EIM data.

Return Value	Meaning
ECONVERT	Data conversion error.
	EIMERR_DATA_CONVERSION (13) (z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.
	EIMERR_CONN_INVAL (54) Connection type is not valid.
	EIMERR_FUNCTION_NOT_SUPPORTED (70) The specified or configured EIM Domain controller does not support this API.
	EIMERR_NOT_SECURE (32) The system is not configured to connect to a secure port. Connection type of EIM_CLIENT_AUTHENTICATION is not valid.
	EIMERR_PARM_REQ (34) Missing required parameter. Please check the API documentation.
	EIMERR_PROTECT_INVAL (22) The protect parameter in EimSimpleConnectInfo is not valid.
	EIMERR_PTR_INVAL (35) Pointer parameter is not valid.
	EIMERR_SSL_REQ (42) The EIM domain controller URL begins with ldaps://, but the SSL information was not specified as a parameter to the EIM API.
	EIMERR_URL_NODN (45) URL has no DN (required).
	EIMERR_URL_NODOMAIN (46) URL has no domain (required).
	EIMERR_URL_NOHOST (47) URL does not have a host.
	EIMERR_URL_NOTLDAP (49) URL does not begin with ldap.
	EIMERR_CREDS_MUST_BE_NULL (58) The connectInfo parameter of the EIM API does not have a NULL value for the creds field in the EimConnectInfo structure.
ENOMEM	Unable to allocate required space.
	EIMERR_NOMEM (27) No memory available. Unable to allocate required space.
ENOTSUP	Connection type is not supported.
	EIMERR_CONN_NOTSUPP (12) Connection type is not supported.
EROFS	LDAP connection is for read-only. Need to connect to master. Use the URL for the master EIM domain controller which is writeable. A writeable connection can also be established by using the EimConnectToMaster API.
	EIMERR_URL_READ_ONLY (50) LDAP connection can be made only to a replica LDAP server. Change the connection information to a writable server and try the request again.
EUNKNOWN	Unexpected exception.
	EIMERR_LDAP_ERR (23) Unexpected LDAP error.
	EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Example

The following example changes the description of the specified EIM domain and enables the use of policy associations for the domain:

```
#include <eim.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
    int            rc;
    char           eimerr[200];
    EimRC        * err;
    EimConnectInfo con;

    char * ldapURL = "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";

    /* Set up connection information           */
    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=admin";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Set up error structure.                 */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    /* Change the description for this domain. */
    if (0 != (rc = eimChangeDomain(ldapURL,
                                   con,
                                   EIM_DOMAIN_DESCRIPTION,
                                   "This is the new description",
                                   EIM_CHG,
                                   err)))
        printf("Change domain description failed; return code = %d", rc);

    /* Set up connection information for second call */
    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=admin";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* reset error structure.                */
    memset(eimerr,0x00,200);
    err->memoryProvidedByCaller = 200;

    /* Enable Policy Associations for this domain. */
    if (0 != (rc = eimChangeDomain(ldapURL,
                                   con,
                                   EIM_DOMAIN_POLICY_ASSOCIATIONS,
                                   NULL,
                                   EIM_ENABLE,
                                   err)))
        printf("Enable policy associations failed; return code = %d", rc);


    return 0;
}
```

eimChangeIdentifier

Purpose

Modifies an existing EIM identifier.

Format

```
#include <eim.h>


int eimChangeIdentifier(EimHandle              * eim,
                        EimIdentifierInfo      * idName,
                        enum EimIdentifierAttr   attrName,
                        char                   * attrValue,
                        enum EimChangeType       changeType,
                        EimRC                  * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

idName
(Input) A structure that contains the name for this identifier.

The following special characters are not allowed in identifier names.

```
, = + < > # ; \ *
```

The layout of the EimIdentifierInfo structure follows:

```
enum EimIdType {
       EIM_UNIQUE_NAME,
       EIM_ENTRY_UUID,
       EIM_NAME
};


typedef struct EimIdentifierInfo
{
    union {
        char       * uniqueName;
        char       * entryUUID;
        char       * name;
    } id;
    enum EimIdType        idtype;
} EimIdentifierInfo;
```

`idtype`
The `idtype` in the EimIdentifierInfo structure indicates which identifier name has been provided. EIM_UNIQUE_NAME finds at most one matching identifier. EIM_NAME results in an error if your EIM domain has more than one identifier containing the same name.

attrName
The attribute to be updated. Valid values are:

EIM_IDENTIFIER_DESCRIPTION (0)
Change the identifier description. Valid *changeType* is EIM_CHG (0).

EIM_IDENTIFIER_NAME (1)
Add or remove a name attribute for this identifier. Valid changeType can be:

- EIM_ADD (1)
- EIM_RMV (2)

EIM_IDENTIFIER_ADDL_INFO (2)
Add or remove an additional information attribute for this identifier. Additional information is user-defined data. Valid *changeType* can be:

- EIM_ADD (1)
- EIM_RMV (2)

attrValue
(Input) The new value for the attribute. This value can be a NULL string (for example, ″″).

changeType
(Input) The type of change to make. On z/OS, this can be one of the following:

EIM_CHG (0)
EIM sets the attribute to the new value. EIM creates the attribute if it does not already exist.

EIM_ADD (1)
EIM adds the attribute and its value to the identifier. EIM creates the attribute if it does not already exist.

EIM_RMV (2)
EIM removes *attrValue* from the attribute in the identifier entry. EIM removes the attribute itself from the entry if no values remain for the attribute. To remove the entire attribute, use NULL for *attrValue*.

The *attrName* parameter indicates the type allowed for each attribute.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddIdentifier” on page 179
- “eimGetAssociatedIdentifiers” on page 254
- “eimListIdentifiers” on page 305
- “eimRemoveIdentifier” on page 364

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM identifiers administrator

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBADNAME	Identifier name is not valid or insufficient access to EIM data.	
	EIMERR_IDNAME_AMBIGUOUS (20)	More than one EIM identifier was found that matches the requested Identifier name.
	EIMERR_NOIDENTIFIER (25)	EIM identifier not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_ATTR_INVAL (5)	Attribute name is not valid.
	EIMERR_CHGTYPE_INVAL (9)	This change type is not valid with the requested attribute. Please check the API documentation.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_IDNAME_TYPE_INVAL (52)	The EimIdType value is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete the operation. Use the eimConnectToMaster API to get a write connection.

Return Value	Meaning	
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates changing an EIM identifier description:

```
#include <eim.h>
.
 .
 .
    int           rc;
    char          eimerr[200];
    EimRC       * err;
    EimHandle     handle;
    EimIdentifierInfo idInfo;

    /* Set up error structure.                */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;
 .
 .
 .

    /* Set up identifier information          */
    idInfo.idtype = EIM_UNIQUE_NAME;
    idInfo.id.uniqueName = "Mary Smith";

    /* Change the description of the identifier */
    rc = eimChangeIdentifier(&handle,
                             &idInfo,
                             EIM_IDENTIFIER_DESCRIPTION,
                             "This is a new description",
                             EIM_CHG,
                             err);
 .
 .
 .
```

eimChangeRegistry

Purpose

Changes the attribute of a registry participating in the EIM domain.

Format

```
#include <eim.h>


int eimChangeRegistry(EimHandle            * eim,
                      char                 * registryName,
                      enum EimRegistryAttr   attrName,
                      char                 * attrValue,
                      enum EimChangeType     changeType,
                      EimRC                * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

registryName
(Input) The name of the registry to change. Registry names are case-independent (meaning, not case-sensitive).

The following special characters are not allowed in registry names:

```
, = + < > # ; \ *
```

attrName
(Input) The attribute to update. Valid values are:

EIM_REGISTRY_DESCRIPTION (0)
Change the registry description. Valid *changeType* is EIM_CHG (0).

EIM_REGISTRY_LABELEDURI (1)
Change the URI for the system registry. Valid *changeType* is EIM_CHG (0).

EIM_REGISTRY_MAPPING_LOOKUP (2)
Change the indicator for whether or not the registry supports mapping lookup operations. By default, mapping lookup operations are supported. Valid *changeType* is EIM_ENABLE (3) or EIM_DISABLE (4).

EIM_REGISTRY_POLICY_ASSOCIATIONS (3)
Change the indicator for whether or not the registry supports policy associations in a mapping lookup. By default, the policy associations are not supported. Valid *changeType* is EIM_ENABLE (3) or EIM_DISABLE (4).

attrValue
(Input) The new value for the attribute. The value can be a NULL string (for example, ″″). Specifically, if the attribute being changed is EIM_REGISTRY_MAPPING_LOOKUP or EIM_REGISTRY_POLICY_ASSOCIATIONS, this value must be NULL.

changeType
(Input) The type of change to make. On z/OS, this could be:

EIM_CHG (0)
EIM sets the attribute to the new value (0).

EIM_ENABLE (3)
Enable the change specified by *attrname*.

EIM_DISABLE (4)
Disable the change specified by *attrname*.

The *attrName* parameter indicates which changeType is allowed for each attribute.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddApplicationRegistry” on page 170
- “eimAddSystemRegistry” on page 190
- “eimListRegistries” on page 317
- “eimRemoveRegistry” on page 374

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM registry *X* administrator

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBADNAME	Registry not found or insufficient access to EIM data.	
	EIMERR_NOREG (28)	EIM registry not found or insufficient access to EIM data.

Return Value	Meaning	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_ATTR_INVAL (5)	Attribute name is not valid.
	EIMERR_CHGTYPE_INVAL (9)	This change type is not valid with the requested attribute. Please check the API documentation.
	EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified or configured EIM Domain controller does not support this API.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master. Use eimConnectToMaster to get a write connection or use the URL for the master EIM domain controller which is writeable.	
	EIMERR_READ_ONLY (36)	This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete this operation. Use eimConnectToMaster to get a write connection.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates changing the description for the registry and enables the use of policy associations for the registry:

```
#include <eim.h>
#include <string.h>
.
.
.
     int           rc;
     char          eimerr[200];
     EimRC       * err;
     EimHandle     handle;
.
```

```
.
.
     /* Set up error structure.                      */
     memset(eimerr,0x00,200);
     err = (EimRC *)eimerr;
     err->memoryProvidedByCaller = 200;

     /* Change the description for this registry     */
     rc = eimChangeRegistry(&handle,
                            "MyAppRegistry",
                            EIM_REGISTRY_DESCRIPTION,
                            "New description",
                            EIM_CHG,
                            err);
.
.
.
     /* reset error structure                        */
     memset(eimerr,0x00,200);
     err->memoryProvidedByCaller = 200;

     /* Enable policy associations for this registry */
     rc = eimChangeRegistry(&handle,
                            "MyAppRegistry",
                            EIM_REGISTRY_POLICY_ASSOCIATIONS,
                            NULL,
                            EIM_ENABLE,
                            err);
.
.
.
```

eimChangeRegistryAlias

Purpose

Adds or removes a registry alias for the defined registry.

Using registry aliases is one way to decouple registry names that developers use from the registry names that administrators choose. Developers who are designing applications know the registry type their application uses and choose the registry alias their program will use. Developers inform the administrator which registry types their applications use and the EIM registry aliases to associate with that registry type. The administrator adds the registry alias to the EIM registry of the appropriate type. The application can use the eimGetRegistryNameFromAlias API; given a registry alias, this API returns the registry name for the entry or entries with that registry alias.

Format

```
#include <eim.h>


int eimChangeRegistryAlias(EimHandle          * eim,
                           char               * registryName,
                           char               * aliasType,
                           char               * aliasValue,
                           enum EimChangeType   changeType,
                           EimRC              * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

registryName
(Input) The name of the registry to change. Registry names are case-independent (meaning, not case-sensitive).

The following special characters are not allowed in registry names:

```
, = + < > # ; \ *
```

aliasType
(Input) A type of alias for this registry. The registry types that EIM provides include the following:

- EIM_ALIASTYPE_DNS ″DNSHostName″
- EIM_ALIASTYPE_KERBEROS ″KerberosRealm″
- EIM_ALIASTYPE_ISSUER ″IssuerDN″
- EIM_ALIASTYPE_ROOT ″RootDN″
- EIM_ALIASTYPE_TCPIP ″TCPIPAddress″
- EIM_ALIASTYPE_LDAPDNSHOSTNAME ″LdapDnsHostName″
- EIM_ALIASTYPE_OTHER ″Other″

To view the eim.h sample, refer to “eim.h” on page 395. Users can define their own registry alias types. See “EIM registry definitions and aliasing” on page 15 for details.

aliasValue
(Input) The value for this alias.

Note: Do not include the asterisk (*) wild card character in names for registry aliases.

changeType
(Input) The type of change to make. This could be add or remove. Use EIM_ADD(1) to add an alias, and EIM_RMV(2) to remove an alias.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimGetRegistryNameFromAlias” on page 265
- “eimListRegistryAliases” on page 325

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM registry *X* administrator

z/OS authorization
No special authorization is necessary.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBADNAME	Registry not found or insufficient access to EIM data.	
	EIMERR_NOREG (28)	EIM registry not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.)Error occurred when converting data between code pages.

Return Value	Meaning	
EINVAL	Input parameter was not valid.	
	EIMERR_CHGTYPE_INVAL (9)	This change type is not valid with the requested attribute. Please check the API documentation.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete the operation. Use the eimConnectToMaster API to get a write connection.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates adding DNS and TCP/IP alias to the registry:

```
#include <eim.h>


.
 .
 .
    int           rc;
    char          eimerr[200];
    EimRC       * err;
    EimHandle     handle;

    /* Set up error structure.                 */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;
 .
 .
 .

     /* Add a dns alias for this registry       */
    rc = eimChangeRegistryAlias(&handle,
                                "MyRegistry",
                                EIM_ALIASTYPE_DNS,
                                "Clueless",
                                EIM_ADD,
                                err);
 .
 .
```

```
.

   /* Add a tcpip address as an alias         */
   rc = eimChangeRegistryAlias(&handle,
                               "MyRegistry",
                               EIM_ALIASTYPE_TCPIP,
                               "254.237.190.239",
                               EIM_ADD,
                               err);
.
.
.
```

eimChangeRegistryUser

Purpose

Changes the attributes for a target registry user.

There are situations when a mapping lookup operation can return more than one user. Applications can choose to use information in the additional information field to distinguish between returned target identities and determine which to use. For example, suppose Joe has two identities in a specific registry X, `joeuser` and `joeadmin`. An application provider can tell the administrator to add additional information, for example, ″appname-admin,″ to the appropriate registry user -- in this case, `joeadmin`. The application can provide this additional information on the lookup APIs, eimGetTargetFromSource and eimGetTargetFromIdentifier.

Format

```
#include <eim.h>


int eimChangeRegistryUser(EimHandle               * eim,
                          char                    * registryName,
                          char                    * registryUserName,
                          enum EimRegistryUserAttr  attrName,
                          char                    * attrValue,
                          enum EimChangeType        changeType,
                          EimRC                   * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

registryName
(Input) The name of the registry that contains this user. Registry names are case-independent (meaning, not case-sensitive).

The following special characters are not allowed in registry names:

```
, = + < > # ; \ *
```

registryUserName
(Input) The name of the user to change in this registry. The registry user name should begin with a non-blank character.

attrName
The attribute to update. Valid values are:

EIM_REGISTRYUSER_DESCRIPTION (0)
Change the registry description. Valid *changeType* is EIM_CHG (0).

EIM_REGISTRYUSER_ADDL_INFO (1)
Add or remove additional information for this user. You can have more than one AdditionalInfo field. Valid *changeType* is EIM_ADD (1) or EIM_RMV (2).

attrValue
(Input) The new value for the attribute. This value can be a NULL string (for example, ″″).

changeType
(Input) The type of change to make. This could be add, remove, or change. On z/OS, this can be one of the following:

EIM_CHG (0)
EIM sets the attribute to the new value. EIM creates the attribute if it does not already exist.

EIM_ADD (1)
EIM adds the attribute and its value to the identifier. EIM creates the attribute if it does not already exist.

EIM_RMV (2)
EIM removes the given attribute value from the attribute in the identifier entry. EIM removes the attribute itself from the entry if no values remain for the attribute. To remove the entire attribute, use NULL for the attribute value.

The *attrName* parameter indicates the type allowed for each attribute.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see "EimRC -- EIM return code parameter for C/C++" on page 164.

Related Information

See the following:

- "eimListRegistryUsers" on page 338

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM registry *X* administrator

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning
0	Request was successful.
EACCES	Access denied. Not enough permissions to access data. **EIMERR_ACCESS (1)** Insufficient access to EIM data.
EBADDATA	eimrc is not valid.

Return Value	Meaning	
EBADNAME	Registry not found or insufficient access to EIM data.	
	EIMERR_NOREG (28)	EIM registry not found or insufficient access to EIM data.
	EIMERR_NOREGUSER (29)	Registry user not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_ATTR_INVAL (5)	Attribute name is not valid.
	EIMERR_CHGTYPE_INVAL (9)	This change type is not valid with the requested attribute. Please check the API documentation.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete the operation. Use the eimConnectToMaster API to get a write connection.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates changing the description and adding additional information for a target registry user.

```
#include <eim.h>


.
.
```

```
.
   int           rc;
   char          eimerr[200];
   EimRC       * err;
   EimHandle     handle;

   /* Set up error structure.                */
   memset(eimerr,0x00,200);
   err = (EimRC *)eimerr;
   err->memoryProvidedByCaller = 200;
.
.
.
   /* Change the registry user's description       */
   rc = eimChangeRegistryUser(&handle,
                              "MyRegistry",
                              "mjjones",
                              EIM_REGISTRYUSER_DESCRIPTION,
                              "cool customer",
                              EIM_CHG,
                              err);

   /* Add additional information to the registry user*/
   rc = eimChangeRegistryUser(&handle,
                              "MyRegistry",
                              "mjjones",
                              EIM_REGISTRYUSER_ADDL_INFO,
                              "security officer",
                              EIM_ADD,
                             err);

   /* Add additional information to the registry user*/
   rc = eimChangeRegistryUser(&handle,
                              "MyRegistry",
                              "mjjones",
                              EIM_REGISTRYUSER_ADDL_INFO,
                              "administrator",
                              EIM_ADD,
                              err);
```

eimConnect

Purpose

Connects to the EIM domain.

Format

```
#include <eim.h>


int eimConnect(EimHandle      * eim,
               EimConnectInfo   connectInfo,
               EimRC          * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns.

connectInfo
(Input) Connect information. This parameter provides the information required to bind to LDAP. If the system is configured to connect to a secure port, EimSSLInfo is required.

For the EIM_SIMPLE connect type, the `creds` field should contain the EimSimpleConnectInfo structure with a binddn and password.

If the connect type is EIM_SIMPLE and you provide no binddn or password, the connection information extracted from the RACF database during the eimCreateHandle API call is used.

Note: Both the bindDn and bindPw must be NULL. Also, the previous call to eimCreateHandle must have been made with a NULL ldapURL in order for eimCreateHandle to have extracted the information from the RACF database. The resulting handle should then be used with eimConnect. If the ldapURL was not NULL, then no information was extracted from the RACF database and a NULL bindDn and bindPw will result in an EIMERR_PARM_REQ error.

EimPasswordProtect determines the level of password protection on the LDAP bind.

EIM_PROTECT_NO (0) The clear-text password is sent on the bind.

EIM_PROTECT_CRAM_MD5 (1)
The protected password is sent on the bind. The server side must support cram-md5 protocol to send the protected password.

EIM_PROTECT_CRAM_MD5_OPTIONAL (2)
The protected password is sent on the bind if the cram-md5 protocol is supported. Otherwise, the clear-text password is sent.

For EIM_KERBEROS, the default logon credentials are used. The `kerberos_creds` field must be NULL.

For EIM_CLIENT_AUTHENTICATION, the `creds` field is ignored. The `ssl` field must point to a valid EimSSLInfo structure. The `keyring` field is required in the EimSSLInfo structure. It can be the name of a System SSL key database file or a RACF keyring name. The `keyring_pw` field is required when the keyring is the

name of a System SSL key database field. The `certificateLabel` field is optional. If it is NULL the default certificate in the keyring is used.

The structure layouts follow:

```
enum EimPasswordProtect {
    EIM_PROTECT_NO,
    EIM_PROTECT_CRAM_MD5,
    EIM_PROTECT_CRAM_MD5_OPTIONAL
};

enum EimConnectType {
    EIM_SIMPLE,
    EIM_KERBEROS,
    EIM_CLIENT_AUTHENTICATION
};


typedef struct EimSimpleConnectInfo
{
     enum EimPasswordProtect protect;
     char * bindDn;
     char * bindPw;
} EimSimpleConnectInfo;

typedef struct EimSSLInfo
{
     char * keyring;
     char * keyring_pw;
     char * certificateLabel;
} EimSSLInfo;


typedef struct EimConnectInfo
{
     enum EimConnectType type;
     union {
         gss_cred_id_t * kerberos;
         EimSimpleConnectInfo simpleCreds;
     } creds;
   EimSSLInfo * ssl;
 } EimConnectInfo;
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimConnectToMaster” on page 220
- “eimCreateHandle” on page 230
- “eimDestroyHandle” on page 239
- “eimGetAttribute” on page 261
- “eimSetAttribute” on page 383

Authorization

z/OS authorization
The calling application can be running in system key or supervisor state or one of the following:

- The RACF user ID of the caller's address space has READ access to the BPX.SERVER profile in the FACILITY class
- The current RACF user ID has READ authority to the IRR.RDCEKEY profile in the FACILITY class

Applications that are not authorized (problem program state and keys) must be program controlled (extattr +p) and the FACILITY class must be active and RACLISTed before the application will be granted authority to use this SAF service.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_CONN_INVAL (54)	Connection type is not valid.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_NOT_SECURE (32)	The system is not configured to connect to a secure port. Connection type of EIM_CLIENT_AUTHENTICATION is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SSL_REQ (42)	The system is configured to connect to a secure port. EimSSLInfo is required.
	EIMERR_CREDS_MUST_BE_NULL (58)	The connectInfo parameter of the EIM API does not have a NULL value for the creds field in the EimConnectInfo structure.
EISCONN	A connection has already been established.	
	EIMERR_CONN (11)	Connection already exists.
EMVSSAFEXTRERR	A connection has already been established.	
	EIMERR_ZOS_R_DCEKEY (6008)	Callable service failed.
	EIMERR_ZOS_R_DCEKEY_BINDPW (6009)	Callable service failed. Bind password is missing.

Return Value	Meaning	
EMVSSAF2ERR	SAF/RACF error	
	EIMERR_ZOS_NO_ACEE (6010)	No task or address space ACEE found.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTSUP	Connection type is not supported.	
	EIMERR_CONN_NOTSUPP (12)	Connection type is not supported.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates connecting to an EIM domain:

```
#include <eim.h>
#include <string.h>


.
.
.
int           rc;
char          eimerr[200];
EimRC       * err;
EimHandle     handle;

EimConnectInfo con;

/* Set up error structure.                */
memset(eimerr,0x00,200);
err = (EimRC *)eimerr;
err->memoryProvidedByCaller = 200;

/* Set up connection information          */
con.type = EIM_SIMPLE;
con.creds.simpleCreds.protect = EIM_PROTECT_NO;
con.creds.simpleCreds.bindDn = "cn=admin";
con.creds.simpleCreds.bindPw = "secret";
con.ssl = NULL;
.
.
.
/* Connect to LDAP URL defined by handle with specified connection credentials */
rc = eimConnect(&handle, con, err);
.
.
.
```

The following example illustrates connecting to an EIM domain using the default Kerberos credential for authentication:

```
#include <eim.h>
#include <string.h>
.
.
.
int rc;
char eimerr [200];
```

```
EimRC *err;
EimHandle handle;
EimConnectInfo con;

/*Set up error structure.*/
memset(eimerr,0x00,200);
err =(EimRC *)eimerr;
err->memoryProvidedByCaller =200;

/*Create new eim handle for a specified ldapURL */
ldapURL ="ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";
rc =eimCreateHandle(&handle,ldapURL,err);
.
.
.
/*Set up connection information */
memset(&con, 0x00, sizeof(con));
con.type =EIM_KERBEROS;

/*Connect to LDAP URL defined in handle with the specified connection credentials*/
rc =eimConnect(&handle,con,err);
.
.
.
```

eimConnectToMaster

Purpose

Connects to the EIM master domain controller. You should use this API if an earlier API invocation returned a referral error (EROFS). A referral error indicates that the current EIM connection is to a replica system. To make updates, you must make an explicit connection to the master system. If the host system is not a replica, then the master information retrieved is the same as the host and port defined in the handle.

Format

```
#include <eim.h>


int eimConnectToMaster(EimHandle      * eim,
                       EimConnectInfo   connectInfo,
                       EimRC          * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns.

connectInfo
(Input) Connect information. This parameter provides the information required to bind to LDAP. If the system is configured to connect to a secure port, EimSSLInfo is required.

For the EIM_SIMPLE connect type, the `creds` field should contain the EimSimpleConnectInfo structure with a binddn and password.

On z/OS, if the connect type is EIM_SIMPLE and you provide no bindDn or bindPw, the connection information extracted from the RACF database during the eimCreateHandle API call is used.

Note: Both the bindDn and bindPw must be NULL. Also, the previous call to eimCreateHandle must have been made with a NULL ldapURL in order for eimCreateHandle to have extracted the information from the RACF database. The resulting handle should then be used with eimConnectToMaster. If the ldapURL was not NULL, then no information was extracted from the RACF database and a NULL bindDn and bindPw will result in an EIMERR_PARM_REQ error.

EimPasswordProtect determines the level of password protection on the LDAP bind.

EIM_PROTECT_NO (0) The clear-text password is sent on the bind.

EIM_PROTECT_CRAM_MD5 (1)
The protected password is sent on the bind. The server side must support cram-md5 protocol to send the protected password.

EIM_PROTECT_CRAM_MD5_OPTIONAL (2)
The protected password is sent on the bind if the cram-md5 protocol is supported.

For EIM_KERBEROS, the default logon credentials are used. The `kerberos_creds` field must be NULL.

For EIM_CLIENT_AUTHENTICATION, the `creds` field is ignored. The `ssl` field must point to a valid EimSSLInfo structure. The `keyring` field is required in the EimSSLInfo structure. It can be the name of a System SSL key database file or a RACF keyring name. The `keyring_pw` field is required when the keyring is the name of a System SSL key database field. The `certificateLabel` field is optional. If it is NULL the default certificate in the keyring is used.

The structure layouts follow:

```
enum EimPasswordProtect {
    EIM_PROTECT_NO,
    EIM_PROTECT_CRAM_MD5,
    EIM_PROTECT_CRAM_MD5_OPTIONAL
};

enum EimConnectType {
    EIM_SIMPLE,
    EIM_KERBEROS,
    EIM_CLIENT_AUTHENTICATION
};


typedef struct EimSimpleConnectInfo
{
     enum EimPasswordProtect protect;
     char * bindDn;
     char * bindPw;
} EimSimpleConnectInfo;

typedef struct EimSSLInfo
{
     char * keyring;
     char * keyring_pw;
     char * certificateLabel;
} EimSSLInfo;


typedef struct EimConnectInfo
{
     enum EimConnectType type;
     union {
         gss_cred_id_t * kerberos;
         EimSimpleConnectInfo simpleCreds;
     } creds;
   EimSSLInfo * ssl;
} EimConnectInfo;
```

eimrc

(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimConnect” on page 215
- “eimCreateHandle” on page 230
- “eimDestroyHandle” on page 239
- “eimGetAttribute” on page 261
- “eimSetAttribute” on page 383

Authorization

z/OS authorization

The calling application can be running in system key or supervisor state or one of the following:

- The RACF user ID of the caller's address space has READ access to the BPX.SERVER profile in the FACILITY class
- The current RACF user ID has READ authority to the IRR.RDCEKEY profile in the FACILITY class

Applications that are not authorized (problem program state and keys) must be program controlled (extattr +p) and the FACILITY class must be active and RACLISTed before the application will be granted authority to use this SAF service.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_CONN_INVAL (54)	Connection type is not valid.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_NOT_SECURE (32)	The system is not configured to connect to a secure port. Connection type of EIM_CLIENT_AUTHENTICATION is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PROTECT_INVAL (22)	The protect parameter in EimSimpleConnectInfo is not valid.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SSL_REQ (42)	The system is configured to connect to a secure port. EimSSLInfo is required.
	EIMERR_CREDS_MUST_BE_NULL (58)	The connectInfo parameter of the EIM API does not have a NULL value for the creds field in the EimConnectInfo structure.

Return Value	Meaning	
EISCONN	A connection has already been established.	
	EIMERR_CONN (11)	Connection already exists.
EMVSSAFEXTRERR	SAF/EXTRACT error.	
	EIMERR_ZOS_R_DCEKEY_BINDPW (6008)	R_DCEKEY callable service failed.
	EIMERR_ZOS_R_DCEKEY (6009)	R_ DCEKEY callable service failed. Bind password is missing.
EMVSSAF2ERR	SAF/RACF error	
	EIMERR_ZOS_NO_ACEE (6010)	No task or address space ACEE found.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again. (z/OS does not return this value.)
ENOTSUP	A connection has already been established.	
	EIMERR_CONN_NOTSUPP (12)	Connection type is not supported.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates connecting to an EIM master domain:

```
#include <eim.h>
#include <string.h>


.
.
.
int            rc;
char           eimerr[200];
EimRC        * err;
EimHandle      handle;
EimConnectInfo con;

/* Set up error structure.                */
memset(eimerr,0x00,200);
err = (EimRC *)eimerr;
err->memoryProvidedByCaller = 200;

/* Set up connection information          */
con.type = EIM_SIMPLE;
con.creds.simpleCreds.protect = EIM_PROTECT_NO;
con.creds.simpleCreds.bindDn = "cn=admin";
con.creds.simpleCreds.bindPw = "secret";
con.ssl = NULL;
.
.
.
```

```
/* Connect to master LDAP URL defined in handle with the specified connection credentials*/
rc = eimConnectToMaster(&handle, con, err);
.
.
.
```

The following example illustrates connecting to an EIM master domain using client authentication, referencing the default digital certificate in a key database file:

```
#include <eim.h>
#include <string.h>
    .
    .
    .
    Int rc;
    Char eimerr [200];
    EimRC *err;
    EimHandle handle;
    EimConnectInfo con;
    EimSSLInfo sslinfo;
    char *ldapURL;

    /*Set up error structure.*/
    memset(eimerr,0x00,200);
    err =(EimRC *)eimerr;
    err->memoryProvidedByCaller =200;

    /*Create new eim handle for a secure SSL host */
    ldapURL ="ldaps://eimsystem:636/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";
    rc =eimCreateHandle(&handle,ldapURL,err);
    .
    .
    .
    /*Set up SSL information */
    sslinfo.keyring ="/u/eimuser/ldapclient.kdb";
    sslinfo.keyring_pw ="secret";
    sslinfo.certificateLabel =NULL;

    /*Set up connection information */
    memset(&con, 0x00, sizeof(con));
    con.type =EIM_CLIENT_AUTHENTICATION;
    con.ssl =&sslinfo;

    /*Connect to master LDAP URL defined in handle with specified connection credentials*/
    rc =eimConnectToMaster(&handle,con,err);
    .
    .
    .
```

eimCreateDomain

Purpose

Creates an EIM domain on the specified EIM domain controller.

Format

```
#include <eim.h>


int eimCreateDomain(char            * ldapURL,
                    EimConnectInfo   connectInfo,
                    char           * description,
                    EimRC          * eimrc)
```

Parameters

ldapURL
(Input) A uniform resource locator (URL) that contains the EIM host information. This parameter is required. This URL has the following format:

```
ldap://host:port/dn
```

or

```
ldaps://host:port/dn
```

host:port
Name of the host on which the EIM domain controller is running. (The port number is optional. If not specified, the default LDAP or LDAPS port will be used.)

dn Distinguished name of the domain to create.

Examples:

```
ldap://systemx:389/ibm-eimDomainName=myEimDomain,o=myCompany,c=us
ldaps://systemy:636/ibm-eimDomainName=thisEimDomain,o=myCompany,c=us
```

Note: In contrast with `ldap`, `ldaps` indicates that this host and port combination uses SSL and TLS.

connectInfo
(Input) Connect information. This parameter provides the information required to bind to LDAP. If the system is configured to connect to a secure port, EimSSLInfo is required.

For the EIM_SIMPLE connect type, the `creds` field should contain the EimSimpleConnectInfo structure with a binddn and password.

EimPasswordProtect determines the level of password protection on the LDAP bind.

EIM_PROTECT_NO (0) The clear-text password is sent on the bind.

EIM_PROTECT_CRAM_MD5 (1)
The protected password is sent on the bind. The server side must support cram-md5 protocol to send the protected password.

EIM_PROTECT_CRAM_MD5_OPTIONAL (2)
The protected password is sent on the bind if the cram-md5 protocol is supported. Otherwise, the clear-text password is sent.

For EIM_KERBEROS, the default logon credentials are used. The `kerberos_creds` field must be NULL.

For EIM_CLIENT_AUTHENTICATION, the `creds` field is ignored. The `ssl` field must point to a valid EimSSLInfo structure. The `keyring` field is required in the EimSSLInfo structure. It can be the name of a System SSL key database file or a RACF keyring name. The `keyring_pw` field is required when the keyring is the name of a System SSL key database field. The `certificateLabel` field is optional. If it is NULL the default certificate in the keyring is used.

The structure layouts follow:

```
enum EimPasswordProtect {
    EIM_PROTECT_NO,
    EIM_PROTECT_CRAM_MD5,
    EIM_PROTECT_CRAM_MD5_OPTIONAL
};

enum EimConnectType {
    EIM_SIMPLE,
    EIM_KERBEROS,
    EIM_CLIENT_AUTHENTICATION
};


typedef struct EimSimpleConnectInfo
{
     enum EimPasswordProtect protect;
     char * bindDn;
     char * bindPw;
} EimSimpleConnectInfo;

typedef struct EimSSLInfo
{
     char * keyring;
     char * keyring_pw;
     char * certificateLabel;
} EimSSLInfo;


typedef struct EimConnectInfo
{
     enum EimConnectType type;
     union {
         gss_cred_id_t * kerberos;
         EimSimpleConnectInfo simpleCreds;
     } creds;
   EimSSLInfo * ssl;
} EimConnectInfo;
```

description
(Input) Textual description for the new EIM domain entry. This parameter can be NULL.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimChangeDomain” on page 194
- “eimDeleteDomain” on page 234

- “eimListDomains” on page 298

Authorization

EIM data
LDAP administrators have the authority to create an EIM domain.

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning
0	Request was successful.
EACCES	Access denied. Not enough permissions to access data. **EIMERR_ACCESS (1)** Insufficient access to EIM data.
EBADDATA	eimrc is not valid.
ECONVERT	Data conversion error. **EIMERR_DATA_CONVERSION (13)** (z/OS does not return this value.) Error occurred when converting data between code pages.
EEXIST	EIM domain already exists. **EIMERR_DOMAIN_EXISTS (14)** EIM domain already exists in EIM.

Return Value	Meaning	
EINVAL	Input parameter was not valid.	
	EIMERR_CHAR_INVAL (21)	A restricted character was used in the domain name. The following special characters are not allowed in domain names: = < > # \ *
	EIMERR_CONN_INVAL (54)	Connection type is not valid.
	EIMERR_NOT_SECURE (32)	The system is not configured to connect to a secure port. Connection type of EIM_CLIENT_AUTHENTICATION is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PROTECT_INVAL (22)	The protect parameter in EimSimpleConnectInfo is not valid.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SSL_REQ (42)	The system is configured to connect to a secure port. EimSSLInfo is required.
	EIMERR_URL_NODN (45)	URL has no DN (required).
	EIMERR_URL_NODOMAIN (46)	URL has no domain (required).
	EIMERR_URL_NOHOST (47)	URL does not have a host.
	EIMERR_URL_NOTLDAP (49)	URL does not begin with ldap.
	EIMERR_CREDS_MUST_BE_NULL (58)	The connectInfo parameter of the EIM API does not have a NULL value for the creds field in the EimConnectInfo structure.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTSUP	Connection type is not supported.	
	EIMERR_CONN_NOTSUPP (12)	Connection type is not supported.
EROFS	LDAP connection is for read-only. Need to connect to master. A writeable connection can be established by using the EimConnectToMaster API.	
	EIMERR_URL_READ_ONLY (50)	LDAP connection can be made only to a replica LDAP server. Change the connection information and try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example creates an EIM domain with the name of `myEIMDomain`. The distinguished name for the domain after it is created will be: ″ibm-eimDomainName=myEIMDomain,o=mycompany,c=us″.

```
#include <eim.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[200];
    EimRC       * err;

    char * ldapURL =
       "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";

    EimConnectInfo con;

    /* Set up connection information           */
    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=admin";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;


    /* Set up error structure.                 */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;
      /* Create a new EIM domain                 */
    if (0 != (rc = eimCreateDomain(ldapURL,
                                   con,
                                   NULL,
                                   err)))
        printf("Create domain error = %d", rc);

    return 0;
}
```

eimCreateHandle

Purpose

Allocates an EimHandle structure, which is used to identify the EIM connection and to maintain per-connection information. The EimHandle structure is passed on subsequent calls to other EIM operations.

Format

```
#include <eim.h>


int eimCreateHandle(EimHandle      * eim,
                    char           * ldapURL,
                    EimRC          * eimrc)
```

Parameters

eim
(Output) The pointer to an EIM handle to return. This handle is input for other EIM APIs.

ldapURL
A NULL parameter indicates the ldapURL information is retrieved from a RACF profile. eimCreateHandle uses the LDAP host name and domain distinguished name stored in the profile to create the ldapURL. The eimCreateHandle retrieves the information from one of the following profiles in this order:

1. The LDAPBIND class profile associated with the caller's user profile
2. The IRR.EIM.DEFAULTS profile in the LDAPBIND class
3. The system default profile, IRR.PROXY.DEFAULTS profile in the FACILITY class

This URL has the following format:

```
ldap://host:port/dn
ldaps://host:port/dn
```

host:port
Name of the host on which the EIM domain controller is running. (The port number is optional. If not specified, the default LDAP or LDAPS port will be used.)

dn Distinguished name of the domain to change.

Examples:

```
ldap://systemx:389/ibm-eimDomainName=myEimDomain,o=myCompany,c=us
ldaps://systemy:636/ibm-eimDomainName=thisEimDomain,o=myCompany,c=us
```

Note: In contrast with `ldap`, `ldaps` indicates that this host and port combination uses SSL and TLS.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see "EimRC -- EIM return code parameter for C/C++" on page 164.

Related Information

See the following:

- "eimConnect" on page 215
- "eimConnectToMaster" on page 220
- "eimDestroyHandle" on page 239
- "eimGetAttribute" on page 261
- "eimSetAttribute" on page 383

Authorization

z/OS authorization

The calling application can be running in system key or supervisor state or one of the following:

- The RACF user ID of the caller's address space has READ authority to the BPX.SERVER profile in the FACILITY class
- The current RACF user ID has READ authority to the IRR.RGETINFO.EIM profile in the FACILITY class

The FACILITY class must be active and RACLISTed before unauthorized (problem program state and keys) will be granted the authority to use this SAF service.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
	EIMERR_URL_NODN (45)	URL has no DN (required).
	EIMERR_URL_NODOMAIN (46)	URL has no domain (required).
	EIMERR_URL_NOHOST (47)	URL does not have a host.
	EIMERR_URL_NOTLDAP (49)	URL does not begin with ldap.

Return Value	Meaning	
EMVSSAFEXTRERR	SAF/RACF EXTRACT error.	
	EIMERR_ZOS_USER_XTR (6002)	RACROUTE REQUEST=EXTRACT error retrieving EIM configuration information from the caller's USER profile.
	EIMERR_ZOS_XTR_EIM (6003)	RACROUTE REQUEST=EXTRACT error retrieving EIM information from a RACF profile.
	EIMERR_ZOS_XTR_PROXY (6005)	RACROUTE REQUEST=EXTRACT error retrieving PROXY information from a RACF profile.
EMVSSAF2ERR	SAF/RACF error.	
	EIMERR_ZOS_XTR_DOMAINDN (6004)	EIM domain distinguished name is missing.
	EIMERR_ZOS_XTR_LDAPHOST (6006)	PROXY LDAP host is missing.
	EIMERR_ZOS_XTR_BINDDN (6007)	PROXY bind distinguished name is missing.
	EIMERR_ZOS_XTR_BINDPW (6018)	PROXY bind password is missing.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOSYS	EIM is not configured	
	EIMERR_NOTCONFIG (30)	(Only z/OS returns this value.) EIM environment is not configured. On z/OS, issue RACF commands to correct the configuration error and then try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates creating an EIM handle with an LDAP URL and using information stored in a RACF profile.

```
#include <eim.h>


   .
 .
 .

    int           rc;
    char          eimerr[200];
    EimRC       * err;
    EimHandle     handle;
    EimHandle     handle2;
    char * ldapURL =
      "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";

     /* Set up error structure.                */
    memset(eimerr,0x00,200);
```

```
   err = (EimRC *)eimerr;
   err->memoryProvidedByCaller = 200;

    /* Create a new eim handle using stored LDAP host and DomainDN in RACF profile */
   rc = eimCreateHandle(&handle, NULL, err);
.
.
.
   /* Create a new eim handle using a specified URL */
   rc = eimCreateHandle(&handle2, ldapURL, err);
.
.
.
```

eimDeleteDomain

Purpose

Deletes the EIM domain information. If there are any registries or identifiers in the domain, then it cannot be deleted.

Format

```
#include <eim.h>


int eimDeleteDomain(char           * ldapURL,
                    EimConnectInfo   connectInfo,
                    EimRC          * eimrc)
```

Parameters

ldapURL
(Input) A uniform resource locator (URL) that contains the EIM host information. This parameter is required. This URL has the following format:

```
ldap://host:port/dn
```

or

```
ldaps://host:port/dn
```

host:port
Name of the host on which the EIM domain controller is running. (The port number is optional. If not specified, the default LDAP or LDAPS port will be used.)

dn Distinguished name of the domain to delete.

Examples:

```
ldap://systemx:389/ibm-eimDomainName=myEimDomain,o=myCompany,c=us
ldaps://systemy:636/ibm-eimDomainName=thisEimDomain,o=myCompany,c=us
```

Note: In contrast with `ldap`, `ldaps` indicates that this host and port combination uses SSL and TLS.

connectInfo
(Input) Connect information. This parameter provides the information required to bind to LDAP. If the system is configured to connect to a secure port, EimSSLInfo is required.

For the EIM_SIMPLE connect type, the `creds` field should contain the EimSimpleConnectInfo structure with a binddn and password.

EimPasswordProtect determines the level of password protection on the LDAP bind.

EIM_PROTECT_NO (0) The clear-text password is sent on the bind.

EIM_PROTECT_CRAM_MD5 (1)
The protected password is sent on the bind. The server side must support cram-md5 protocol to send the protected password.

EIM_PROTECT_CRAM_MD5_OPTIONAL (2)
The protected password is sent on the bind if the cram-md5 protocol is supported. Otherwise, the clear-text password is sent.

For EIM_KERBEROS, the default logon credentials are used. The `kerberos_creds` field must be NULL.

For EIM_CLIENT_AUTHENTICATION, the `creds` field is ignored. The `ssl` field must point to a valid EimSSLInfo structure. The `keyring` field is required in the EimSSLInfo structure. It can be the name of a System SSL key database file or a RACF keyring name. The `keyring_pw` field is required when the keyring is the name of a System SSL key database field. The `certificateLabel` field is optional. If it is NULL the default certificate in the keyring is used.

The structure layouts follow:

```
enum EimPasswordProtect {
    EIM_PROTECT_NO,
    EIM_PROTECT_CRAM_MD5,
    EIM_PROTECT_CRAM_MD5_OPTIONAL
};

enum EimConnectType {
    EIM_SIMPLE,
    EIM_KERBEROS,
    EIM_CLIENT_AUTHENTICATION
};


typedef struct EimSimpleConnectInfo
{
     enum EimPasswordProtect protect;
     char * bindDn;
     char * bindPw;
} EimSimpleConnectInfo;

typedef struct EimSSLInfo
{
     char * keyring;
     char * keyring_pw;
     char * certificateLabel;
} EimSSLInfo;


typedef struct EimConnectInfo
{
     enum EimConnectType type;
     union {
         gss_cred_id_t * kerberos;
         EimSimpleConnectInfo simpleCreds;
     } creds;
   EimSSLInfo * ssl;
} EimConnectInfo;
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimCreateDomain” on page 225
- “eimChangeDomain” on page 194
- “eimListDomains” on page 298

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

z/OS authorization

No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBADNAME	EIM domain not found or insufficient access to EIM data.	
	EIMERR_NODOMAIN (24)	EIM domain not found or insufficient access to EIM data.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_CONN_INVAL (54)	Connection type is not valid.
	EIMERR_NOT_SECURE (32)	The system is not configured to connect to a secure port. Connection type of EIM_CLIENT_AUTHENTICATION is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PROTECT_INVAL (22)	The protect parameter in EimSimpleConnectInfo is not valid.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SSL_REQ (42)	The system is configured to connect to a secure port. EimSSLInfo is required.
	EIMERR_URL_NODN (45)	URL has no DN (required).
	EIMERR_URL_NODOMAIN (46)	URL has no domain (required).
	EIMERR_URL_NOHOST (47)	URL does not have a host.
	EIMERR_URL_NOTLDAP (49)	URL does not begin with ldap.
	EIMERR_CREDS_MUST_BE_NULL (58)	The connectInfo parameter of the EIM API does not have a NULL value for the creds field in the EimConnectInfo structure.

Return Value	Meaning	
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTSAFE	Not safe to delete domain.	
	EIMERR_DOMAIN_NOTEMPTY (15)	Cannot delete a domain when it has registries or identifiers.
ENOTSUP	Connection type is not supported.	
	EIMERR_CONN_NOTSUPP (12)	Connection type is not supported.
EROFS	LDAP connection is for read-only. Need to connect to master. A writeable connection can be established by using the EimConnectToMaster API.	
	EIMERR_URL_READ_ONLY (50)	LDAP connection can be made only to a replica LDAP server. Change the connection information and try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example deletes the specified EIM domain information:

```
#include <eim.h>
#include <string.h>


int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[200];
    EimRC       * err;

    char * ldapURL =
      "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";

    EimConnectInfo con;

    /* Set up connection information           */
    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=admin";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;


    /* Set up error structure.                 */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    /* Delete this domain                      */
    if (0 != (rc = eimDeleteDomain(ldapURL,
                                   con,
```

```
                                        err)))
            printf("Delete domain error = %d", rc);
        return 0;
    }
```

eimDestroyHandle

Purpose

Frees resources associated with the EimHandle and closes connections to the EIM domain controllers. This closes the EIM connection for this handle.

Format

```
#include <eim.h>


int eimDestroyHandle(EimHandle      * eim,
                     EimRC          * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimConnect” on page 215
- “eimConnectToMaster” on page 220
- “eimCreateHandle” on page 230
- “eimGetAttribute” on page 261
- “eimSetAttribute” on page 383

Authorization

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.

Return Value	Meaning	
EINVAL	Input parameter was not valid.	
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
EUNKNOWN	Unexpected exception.	
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates destroying an EIM handle:

```
#include <eim.h>


   .
 .
 .

    int           rc;
    char          eimerr[200];
    EimRC       * err;
    EimHandle   * handle;

    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

 .
 .
 .

    /* Destroy the handle                       */
    rc = eimDestroyHandle(handle, err);
 .
 .
 .
```

eimErr2String

Purpose

Converts the EIM return code structure that an EIM function returns into a NULL-terminated character string that describes the error.

Format

```
#include <eim.h>


 char * eimErr2String(EimRC * eimrc)
```

Parameters

eimrc
(Input) The structure in which to return error code information. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Authorization

z/OS authorization
None.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning
address of error string	Request was successful. (The caller is expected to free the error string.)
NULL	Request was unsuccessful. The eimErr2String sets global errno. The errno can be set by catopen, catgets, catclose, or one of the following values: **EBADDATA** eimrc is not valid. No eimrc structure was provided or the eimrc is not large enough to be an eimrc structure.

Example

The following example converts an EIM RC into an error message and prints it.

```
#include <eim.h>
#include <stdio.h>


    ...

    char            eimerr[150];
    EimRC         * err;
    char          * message;

      ...

     /* Set up error structure.                */
    memset(eimerr,0x00,150);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 150;
```

```
/* Call an EIM API that returns an EimRC...*/

/* Convert the error structure to a message           */
if (NULL == (message = eimErr2String(err)))
   printf("eimErr2String error = %s",strerror(errno));
else
{
   printf("EIM API Error Message: %s",message);
   free(message);
}

  ...
```

eimFormatPolicyFilter

Purpose

Takes unformatted user identity information and generates a policy filter value for use with the eimAddPolicyFilter API.

Format

```
#include <eim.h>

int eimFormatPolicyFilter(EimUserIdentityInfo       * userIdentityInfo,
                          EimPolicyFilterSubsetInfo * subsetInfo,
                          unsigned int                lengthOfListData,
                          EimList                   * listData,
                          EimRC                     * eimrc)
```

Parameters

userIdentityInfo

(Input) The user identity information from which to generate policy filter values. This structure contains information about the user identity. For EIM_DER_CERT (0) or EIM_BASE64_CERT (1) user identity type, the userIdentityInfo field must contain an EimCertificate structure. For EIM_CERT_INFO (2) user identity type, the userIdentityInfo field must contain an EimCertificateInfo structure.

The structure layouts follow:

```
enum EimUserIdentityType {
       EIM_DER_CERT,                  /* Entire X.509 public key
                                         certificate in ASN.1 DER encoding,
                                         Public Key Cryptography
                                         Standard 6 (PKCS-6) or PKCS-7 format.    */
       EIM_BASE64_CERT,               /* Base 64 encoded version of the
                                         entire X.509 public key
                                         certificate in ASN.1 DER
                                         encoding, Public Key
                                         Cryptography Standard 6 (PKCS-6) or PKCS-7
                                         format.          */
       EIM_CERT_INFO                  /* Components of the certificate. */
   };
   typedef struct EimCertificateInfo
   {
       char        * issuerDN;        /* The issuer DN.                */
       char        * subjectDN;       /* The subject DN.               */
       void char   * publicKey;       /* The public key info structure (may be NULL).  */
 unsigned int  publicKeyLen;     /* Length of public key info structure (may be 0)*/
   } EimCertificateInfo;
 typedef struct EimCertificate
 {
     char        * certData;        /* The certificate data          */
     unsigned int certLength;       /* The length of the certificate
                                       data.                         */
 } EimCertificate;
 typedef struct EimUserIdentityInfo
 {
     enum EimUserIdentityType type;
     union {
           EimCertificateInfo certInfo;
           EimCertificate  cert;
     } userIdentityInfo;
 } EimUserIdentityInfo;
```

subsetInfo

(Input) The information used to subset the policy filter values that are formatted.

If NULL is specified, then all possible return values are returned for the specified user identity. If this parameter is not NULL, then the results returned are based on the subset information. For EIM_BASE64_CERT (0), EIM_DER_CERT (1), or EIM_CERT_INFO (2) user identity type, the subset field must contain an EimCertPolicyFilterSubsetInfo structure.

The structure layouts follow:

```
typedef struct EimCertPolicyFilterSubsetInfo
{
    char * subjectFilter; /* Subject filter value. */
    char * issuerFilter; /* Issuer filter value. */
} EimCertPolicyFilterSubsetInfo;
typedef struct EimPolicyFilterSubsetInfo
{
    union {
        EimCertPolicyFilterSubsetInfo certFilter;
    } subset;
} EimPolicyFilterSubsetInfo;
```

lengthOfListData

(Input) The number of bytes provided by the caller for the listData parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The minimum size required is 20 bytes.

listData

(Output) A pointer to the EimList structure, which contains information about the returned data. The data takes the form of a linked list of EimPolicyFilterValue structures.

The EimList structure has the following layout:

```
 typedef struct EimList
   {
       unsigned int bytesReturned;     /* Number of bytes actually returned
                                          by the API                       */
       unsigned int bytesAvailable;    /* Number of bytes of available data
                                          that could have been returned by
                                          the API                          */
       unsigned int entriesReturned;   /* Number of entries actually
                                       returned by the API              */
       unsigned int entriesAvailable;  /* Number of entries available to be
                                          returned by the API              */
       unsigned int firstEntry;        /* Displacement to the first linked
                                          list entry. This byte offset is
                                          relative to the start of the
                                          EimList structure.               */
   } EimList;
```

The EimPolicyFilterValue structure has the following layout:

```
typedef struct EimPolicyFilterValue
{
   unsigned int nextEntry; /* Displacement to next entry. This
                              byte offset is relative to the
                              start of this structure */
   EimListData filterValue; /* Generated policy filter value. */
} EimPolicyFilterValue;
```

The EimListData structure has the following layout:

```
typedef struct EimListData
{
    unsigned int length; /* Length of data */
    unsigned int disp;  /*  Displacement to data. This byte
                            offset is relative to the start of
```

```
                                    the parent structure; that is, the
                                    structure containing this
                                    structure. */
    } EimListData;
```

eimrc
(Input) The structure in which to return error code information. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddPolicyFilter” on page 187
- “eimRemovePolicyFilter” on page 371
- “eimListPolicyFilters” on page 312

Authorization

No authorization is required.

Return Values

Return Value	Meaning
0	Request was successful.
EACCES	Access denied. Not enough permissions to access data.
EBADDATA	eimrc is not valid.
ECONVERT	Data conversion error. **EIMERR_DATA_CONVERSION (13)** (z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid. **EIMERR_ASSOC_TYPE_INVAL (4)** Association type is not valid. **EIMERR_EIMLIST_SIZE (16)** Length of EimList is not valid. EimList must be at least 20 bytes in length. **EIMERR_PARM_REQ (34)** Missing required parameter. Please check the API documentation. **EIMERR_PTR_INVAL (35)** (z/OS does not return this value.) Pointer parameter is not valid. **EIMERR_SPACE (41)** Unexpected error accessing parameter. **EIMERR_USER_IDENTITY_TYPE_INVAL (63)** User identity type is not valid. **EIMERR_CERTIFICATE_INVAL (67)** Certificate data is not valid.
ENOMEM	Unable to allocate required space. **EIMERR_NOMEM (27)** No memory available. Unable to allocate required space.
EUNKNOWN	Unexpected exception. **EIMERR_UNKNOWN (44)** Unknown error or unknown system state.

Example

The following example generates certificate policy filter values.

```
#include <eim.h>
#include <stdio.h>
#include <stddef.h>
#include <string.h>
#include <sys/stat.h>
#include <errno.h>

void printListResults(EimList * list);

void printListData(char * fieldName,
                   void * entry,
                   int    offset);

int main (int argc, char *argv[])
{
    int                   rc;
    char                  eimerr[250];
    EimRC              * err;
    EimHandle          * handle;
    EimUserIdentityInfo  idInfo;
    char                  listData[4000];
    EimList            * list = (EimList *)listData;

    FILE               * certFile;
    char               * certBuf;
    int                   certLen;
    struct stat           info;

    /* If no certificate file specified as parameter,    */
    /* print usage message and exit                      */
    if (argc < 2) {
       printf("Usage: %s <cert-file-name>\n", argv[0]);
       return 1;
    } else {
       /* Get certificate file statistics. quit if fails */
       if (stat(argv[1], &info) != 0) {
          printf("stat failed for cert file=[%s]\n", argv[1]);
          return 1;
       }
       /* Make sure certificate file is in fact a file   */
       if (S_ISREG(info.st_mode) == 0) {
          printf("cert file=[%s] is not a regular file\n", argv[1]);
          return 1;
       }
    }

    /* Make sure the certificate file contains data      */
    if (info.st_size == 0) {
       printf("cert file %s contains no data\n", argv[1]);
       return 1;
    }

    /* Obtain storage to contain certificate data        */
    certBuf = (char *)malloc(info.st_size);
    if (certBuf == NULL) {
       printf("Failed to get %d bytes data for cert buffer\n"
             , info.st_size);
       return 1;
    }

    /* Open the certificate file, quit if fails*/
    certFile = fopen(argv[1], "r");
    if (certFile == NULL) {
        printf("Unable to open %s: %s\n",
```

```
                argv[1], strerror(errno));
            free(certBuf);
            return 1;
        }

        /* Read the certificate file, quit if fails*/
        certLen = fread(certBuf, 1, info.st_size, certFile);
        if (certLen == 0) {
            printf("Unable to read %s: %s\n",
                   argv[1], strerror(errno));
            fclose(certFile);
            free(certBuf);
            return 1;
        }
        fclose(certFile);

        /* Set up error structure.                  */
        memset(eimerr,0x00,250);
        err = (EimRC *)eimerr;
        err->memoryProvidedByCaller = 250;

        /* Get user identity information.           */
        idInfo.type = EIM_DER_CERT;
        idInfo.userIdentityInfo.cert.certLength = certLen;
        idInfo.userIdentityInfo.cert.certData = certBuf;

        /* Format EIM Policy Filter                 */
        /* This call will return all possible       */
        /* certificate policy filter values.        */
        if (0 != (rc = eimFormatPolicyFilter(&idInfo,
                                             NULL,
                                             4000,
                                             list,
                                             err)))
        {
            printf("Format Policy Filter failed; return code = %d", rc);
            free(certBuf);
            return -1;
        }
        /* Print the results                        */
        printListResults(list);

        free(certBuf);

        return 0;

    }

    void printListResults(EimList * list)
    {
        int i;
        EimPolicyFilterValue * entry;

        printf("___________\n");
        printf("   bytesReturned    = %d\n", list->bytesReturned);
        printf("   bytesAvailable   = %d\n", list->bytesAvailable);
        printf("   entriesReturned  = %d\n", list->entriesReturned);
        printf("   entriesAvailable = %d\n", list->entriesAvailable);
        printf("\n");
        entry = (EimPolicyFilterValue *)((char *)list + list->firstEntry);
        for (i = 0; i < list->entriesReturned; i++)
        {
            printf("\n");
            printf("===============\n");
            printf("Entry %d.\n", i);

            /* Print out results */
```

```
        printListData("Policy Filter Value",
                      entry,
                      offsetof(EimPolicyFilterValue, filterValue));
        /* advance to next entry */
        entry = (EimPolicyFilterValue *)((char *)entry + entry->nextEntry);
    }
    printf("\n");

}

void printListData(char * fieldName,
                   void * entry,
                   int offset)
{
    EimListData * listData;
    char        * data;
    int           dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}
```

eimFormatUserIdentity

Purpose

Takes unformatted user identity information and formats it for use with other EIM functions.

Format

```
#include <eim.h>

int eimFormatUserIdentity(
                enum EimUserIdentityFormatType formatType,
                     EimUserIdentityInfo *     userIdentityInfo,
                     unsigned int              lengthOfUserIdentity,
                     EimUserIdentity *         userIdentity,
                     EimRC *                   eimrc)
```

Parameters

formatType
(Input) How to format the user identity. Choices for input are:

EIM_REGISTRY_USER_NAME (0)
Indicates the user identity information be used to form a registry user name that may then be used as a registryUserName on other EIM APIs. For certificates or certificate information, the registry user name is formed from the subject distinguished name (SDN), issuer distinguished name (IDN), and a hash of the public key information. The registry user name takes this format:

```
<SDN>subject-DN</SDN><IDN>issuer-DN</IDN><HASH_VAL>hash-value</HASH_VAL>
```

userIdentityInfo
(Input) The user identity information to format. For EIM_DER_CERT (0) or EIM_BASE64_CERT (1) user identity type, the userIdentityInfo field must contain an EimCertificate structure. For EIM_CERT_INFO (2) user identity type, the userIdentityInfo field must contain an EimCertificateInfo structure.

The structure layouts follow:

```
enum EimUserIdentityType {
       EIM_DER_CERT,                  /* Entire X.509 public key
                                         certificate in ASN.1 DER encoding,
                                         Public Key Cryptography
                                         Standard 6 (PKCS-6) or PKCS-7 format.    */
       EIM_BASE64_CERT,               /* Base 64 encoded version of the
                                         entire X.509 public key
                                         certificate in ASN.1 DER
                                         encoding, Public Key
                                         Cryptography Standard 6 (PKCS-6) or PKCS-7
                                         format.
       EIM_CERT_INFO                  /* Components of the certificate. */
   };
   typedef struct EimCertificateInfo
   {
       char        * issuerDN;        /* The issuer DN.                */
       char        * subjectDN;       /* The subject DN.               */
       void        * publicKey;       /* The public key info structure.  */
   unsigned int  publicKeyLen;     /* Length of public key info structure.*/
} EimCertificateInfo;
typedef struct EimCertificate
   {
       unsigned int certLength;       /* The length of the certificate
                                         data.                         */
```

```
       char        * certData;         /* The certificate data           */
   } EimCertificate;
typedef struct EimUserIdentityInfo
{
  enum EimUserIdentityType type;
  union {
     EimCertificateInfo certInfo;
     EimCertificate  cert;
  } userIdentityInfo;
} EimUserIdentityInfo;
```

If the userIdentityInfo field contains an EimCertificateInfo structure, the issuerDN and subjectDN fields must contain valid DN strings (for example *CN=John D. Smith,OU=Sales,O=IBM,L=Rochester,ST=Min,C=US*). The PublicKey field must contain the full DER encoded public key information structure, and the publicKeyLen field must contain the length of that structure.

For EIM_DER_CERT (0) certifications, certData must point to a buffer containing the DER encoded cert and the certLength field must contain the length of the certificate. The length specified in certLength will be verified against the length encoded in the certificate.

For EIM_BASE64_CERT (1) certificates, certData must point to a buffer containing the base64 encoded certificate with or without the BEGIN - END tags removed and certLength must contain the length of certData.

lengthOfUserIdentity
(Input) The number of bytes provided by the caller for the formatted user identify. The size required is calculated as: the length of the subject DN, plus the length of the issuer DN, plus 40 bytes for the hash value, plus 43 bytes for the SDN, IDN and HASH_VAL tags, plus 16 bytes for the other elements of the EimUserIdentity structure. The minimum size is 16 bytes.

userIdentity
(Output) A pointer to the data to be returned. The API will return as much data as the space allows.

EimUserIdentity has the following structure:

```
typedef struct EimUserIdentity
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API.                      */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API.                         */
    EimListData  userIdentity;      /* User identity                    */
} EimUserIdentity;
```

EimListData has the following structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                   */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure; that is, the
                                       structure containing this
                                       structure.                       */
} EimListData;
```

eimrc
(Input) The structure in which to return error code information. If the return

value is not 0, eimrc is set with additional information. This parameter may be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddAssociation” on page 174

Authorization

No authorization is required.

Return Values

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
	EIMERR_USER_IDENTITY_TYPE_I NVAL (63)	User identity type is not valid.
	EIMERR_USER_IDENTITY_SIZE (64)	User identity length is not valid.
	EIMERR_USER_IDENTITY_FORMAT_TYPE_INVAL (65)	User identity format type is not valid.
	EIMERR_CERTIFICATE_INVAL (67)	Certificate data is not valid.
EMVSERR	An MVS environment or internal error has occurred.	
	EIMERR_ZOS_DATA_CONVERSION (6011)	(only z/OS returns this value.) Error occurred when converting data between code pages.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
EUNKNOWN	Unexpected exception.	
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example formats the user identity and adds an association.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <errno.h>

int main (int argc, char *argv[])
{
    int                 rc;
    char                eimerr[250];
    EimRC             * err;
    EimHandle         * handle;
    EimIdentifierInfo   id;
    EimUserIdentityInfo idInfo;
    char                rtnData[4000];
    EimUserIdentity   * fmtData = (EimUserIdentity *)rtnData;

    FILE              * certFile;
    char              * certBuf;
    int                 certLen;
    struct stat         info;

    /* If no certificate file specified as parameter,    */
    /* print usage message and exit                      */
    if (argc < 2) {
       printf("Usage: %s <cert-file-name>\n", argv[0]);
       return 1;
    } else {
       /* Get certificate file statistics. quit if fails */
       if (stat(argv[1], &info) != 0) {
          printf("stat failed for cert file=[%s]\n", argv[1]);
          return 1;
       }
       /* Make sure certificate file is in fact a file   */
       if (S_ISREG(info.st_mode) == 0) {
          printf("cert file=[%s] is not a regular file\n", argv[1]);
          return 1;
       }
    }

    /* Make sure the certificate file contains data      */
    if (info.st_size == 0) {
       printf("cert file %s contains no data\n", argv[1]);
       return 1;
    }

    /* Obtain storage to contain certificate data        */
    certBuf = (char *)malloc(info.st_size);
    if (certBuf == NULL) {
       printf("Failed to get %d bytes data for cert buffer\n"
             , info.st_size);
       return 1;
    }

    /* Open the certificate file, quit if fails*/
    certFile = fopen(argv[1], "r");
    if (certFile == NULL) {
        printf("Unable to open %s: %s\n",
               argv[1], strerror(errno));
        free(certBuf);
        return 1;
    }

    /* Read the certificate file, quit if fails*/
    certLen = fread(certBuf, 1, info.st_size, certFile);
    if (certLen == 0) {
```

```
        printf("Unable to read %s: %s\n",
               argv[1], strerror(errno));
        fclose(certFile);
        free(certBuf);
        return 1;
    }
    fclose(certFile);

    /* Set up error structure. */
    memset(eimerr,0x00,250);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 250;

    /* Get user identity information. */
    idInfo.type = EIM_DER_CERT;
    idInfo.userIdentityInfo.cert.certLength = certLen;
    idInfo.userIdentityInfo.cert.certData = certBuf;

    /* Format user identity */
    if (0 != (rc = eimFormatUserIdentity(EIM_REGISTRY_USER_NAME,
                                         &idInfo,
                                         4000,
                                         fmtData,
                                         err)))
    {
       printf("Format user identity error = %d\n", rc);
       free(certBuf);
       return -1;
    } else {
       printf("Formated user identity: %s\n"
             ,(char *)fmtData + fmtData->userIdentity.disp );
    }

    free(certBuf);

    return 0;
}
```

eimGetAssociatedIdentifiers

Purpose

Returns a list of the identifiers. Given a registry name and registry user name within that user registry, this API returns the EIM identifier associated with it. It is possible that more than one person is associated with a specific user name. This occurs when users share identities (and possibly passwords) within a single instance of a user registry. While this practice is not condoned, it does happen. This creates an ambiguous result.

Format

```
#include <eim.h>


int eimGetAssociatedIdentifiers(EimHandle               * eim,
                                enum EimAssociationType   associationType,
                                char                    * registryName,
                                char                    * registryUserName,
                                unsigned int              lengthOfListData,
                                EimList                 * listData,
                                EimRC                   * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

associationType
(Input) The type of association to retrieve. Valid values are:

EIM_ALL_ASSOC (0)
Retrieve all associations.

EIM_TARGET (1)
Retrieve target associations.

EIM_SOURCE (2)
Retrieve source associations.

EIM_SOURCE_AND_TARGET (3)
Retrieve source and target associations.

EIM_ADMIN (4)
Retrieve administrative associations.

registryName
(Input) The registry name for the lookup. If this string has a null value, the API uses the system default local registry name from the instorage copy of the registry name. Registry names are case-independent (meaning, not case-sensitive).

The following special characters are not allowed in registry names:

```
, = + < > # ; \ *
```

registryUserName
(Input) The registry user name for the lookup.

lengthOfListData
(Input) The number of bytes provided by the caller for the listData parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData

structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The minimum size required is 20 bytes.

listData

(Output) A pointer to the data to return. The EimList structure contains information about the returned data. The data returned is a linked list of EimIdentifier structures. The firstEntry is used to get to the first EimIdentifier structure in the linked list. The number of complete EimIdentifier structures is returned in entriesReturned. The bytesReturned variable has the number of bytes the API used for the returned entries. If the number of entries returned is less than the number of entries available, the returned data contains as many complete EimIdentifier structures as will fit. It can also contain a partial EimIdentifier structure.The EimList structure follows:

```
typedef struct EimList
   {
       unsigned int bytesReturned;     /* Number of bytes actually returned
                                          by the API                      */
       unsigned int bytesAvailable;    /* Number of bytes of available data
                                          that could have been returned by
                                          the API                         */
       unsigned int entriesReturned;   /* Number of entries actually
                                          returned by the API             */
       unsigned int entriesAvailable;  /* Number of entries available to be
                                          returned by the API             */
       unsigned int firstEntry;        /* Displacement to the first linked
                                          list entry. This byte offset is
                                          relative to the start of the
                                          EimList structure.              */
   } EimList;
```

The EimIdentifier structure follows:

```
    typedef struct EimIdentifier
   {
       unsigned int nextEntry;         /* Displacement to next entry.  This
                                          byte offset is relative to the
                                          start of this structure         */
       EimListData uniquename;         /* Unique name                     */
       EimListData description;        /* Description                     */
       EimListData entryUUID;          /* UUID                            */
       EimSubList  names;              /* EimIdentifierName sublist       */
       EimSubList  additionalInfo;     /* EimAddlInfo sublist             */
   } EimIdentifier;
```

Identifiers might have defined several name attributes as well as several additional information attributes. In the EimIdentity structure, the name EimSubList gives addressability to a linked list of EimIdentifierName structures:

```
    typedef struct EimIdentifierName
    {
        unsigned int nextEntry;         /* Displacement to next entry.  This
                                           byte offset is relative to the
                                           start of this structure         */
        EimListData name;               /* Name                            */
    } EimIdentifierName;
```

The additionalInfo EimSubList gives addressability to a linked list of EimAddlInfo structures. The EimAddlInfo structure follows:

```
   typedef struct EimAddlInfo
   {
       unsigned int nextEntry;         /* Displacement to next entry.  This
```

```
                                        byte offset is relative to the
                                        start of this structure          */
    EimListData addlInfo;              /* Additional info                */
} EimAddlInfo;
```

The EimSubList structure follows:

```
typedef struct EimSubList
{
    unsigned int listNum;              /* Number of entries in the list  */
    unsigned int disp;                 /* Displacement to sublist. This
                                        byte offset is relative to the
                                        start of the parent structure, i.e.
                                        the structure containing this
                                        structure.                       */
} EimSubList;
```

The EimListData structure follows:

```
typedef struct EimListData
{
    unsigned int length;               /* Length of data                 */
    unsigned int disp;                 /* Displacement to data.  This byte
                                        offset is relative to the start of
                                        the parent structure, i.e. the
                                        structure containing this
                                        structure.                       */
} EimListData
```

eimrc

(Input/output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddIdentifier” on page 179
- “eimChangeIdentifier” on page 199
- “eimListIdentifiers” on page 305
- “eimRemoveIdentifier” on page 364

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM identifiers administrator
- EIM mapping-lookup authority
- EIM registry *X* administrator

The returned list contains only the information that the user has authority to access.

z/OS authorization

The calling application can be running in system key or supervisor state or one of the following:

- The RACF user ID of the caller's address space has READ authority to the BPX.SERVER profile in the FACILITY class
- The current RACF user ID has READ authority to the IRR.RGETINFO.EIM profile in the FACILITY class

The FACILITY class must be active and RACLISTed before unauthorized (problem program state and keys) will be granted the authority to use this SAF service.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
EBADNAME	Registry not found or insufficient access to EIM data.	
	EIMERR_NOREG (28)	EIM registry not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_ASSOC_TYPE_INVAL (4)	Association type is not valid.
	EIMERR_EIMLIST_SIZE (16)	Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.

Return Value	Meaning	
ENOSYS	EIM is not configured.	
	EIMERR_NOTCONFIG (30)	(Only z/OS returns this value.) EIM environment is not configured. On z/OS, issue RACF commands to correct the configuration error and then try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example lists all of the identiifers associated with the registry, MyRegistry, and a user of carolb.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>


void printListResults(EimList * list);
void printSubListData(char * fieldName, void * entry, int offset);
void printListData(char * fieldName, void * entry, int offset);

int main(int argc, char *argv[])
{
    int            rc;
    char           eimerr[200];
    EimRC        * err;
    EimHandle      handle;
    EimConnectInfo con;
    char         * ldapHost =
      "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";
    char           listData[1000];
    EimList      * list = (EimList * ) listData;

    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=admin";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Create handle with specified LDAP URL    */
    if (0 != (rc = eimCreateHandle(&handle,
                                   ldapHost,
                                   err))) {
        printf("Create handle error = %d\n", rc);
        return -1;
    }

    /* Connect with specified credentials       */
    if (0 != (rc = eimConnect(&handle,
                              con,
```

```
                            err))) {
        printf("Connect error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    /* Get associated identifiers               */
    if (0 != (rc = eimGetAssociatedIdentifiers(&handle,
                                               EIM_ALL_ASSOC,
                                               "MyRegistry",
                                               "carolb",
                                               1000,
                                               list,
                                               err)))
    {
        printf("Get Associated Identifers error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    /* Destroy the handle                       */
    rc = eimDestroyHandle(&handle, err);

    return 0;
}

void printListResults(EimList * list)
{
    int i;
    EimIdentifier * entry;

    printf("___________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimIdentifier *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("===============\n");
        printf("Entry %d.\n", i);

        /* Print out results */
        printListData("Unique name",
                      entry,
                      offsetof(EimIdentifier, uniquename));
        printListData("description",
                      entry,
                      offsetof(EimIdentifier, description));
        printListData("entryUUID",
                      entry,
                      offsetof(EimIdentifier, entryUUID));
        printSubListData("Names",
                         entry,
                         offsetof(EimIdentifier, names));
        printSubListData("Additional Info",
                         entry,
                         offsetof(EimIdentifier, additionalInfo));
        /* advance to next entry */
        entry = (EimIdentifier *)((char *)entry + entry->nextEntry);
    }
```

```
    printf("\n");

}
void printSubListData(char * fieldName, void * entry, int offset)
{
    int i;
    EimSubList * subList;
    EimAddlInfo * subentry;

    /* Address the EimSubList object */
    subList = (EimSubList *)((char *)entry + offset);

    if (subList->listNum > 0)
    {
        subentry = (EimAddlInfo *)((char *)entry + subList->disp);
        for (i = 0; i < subList->listNum; i++)
        {
            /* Print out results */
            printListData(fieldName,
                          subentry,
                          offsetof(EimAddlInfo, addlInfo));
            /* advance to next entry */
            subentry = (EimAddlInfo *)((char *)subentry +
                                       subentry->nextEntry);
        }
    }
}

void printListData(char * fieldName, void * entry, int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}
```

eimGetAttribute

Purpose

Gets attributes for this EIM handle. If the host system for the EIM domain is a replica LDAP server, the handle master attributes contain the host system information for the master LDAP server. If the host system for the EIM domain is a master (meaning it is writable) LDAP server, the HOST, PORT and SECPORT handle attributes and master handle attributes have the same values.

Format

```
#include <eim.h>


int eimGetAttribute(EimHandle          * eim,
                    enum EimHandleAttr   attrName,
                    unsigned int         lengthOfEimAttribute,
                    EimAttribute       * attribute,
                    EimRC              * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns.

attrName
(Input) The name of the attribute to retrieve. The following values are valid:

EIM_HANDLE_CCSID (0) (z/OS does not support this value.) This is the coded character set identifier (CCSID) of character data that the caller of EIM APIs passes with the specified EimHandle. The returned field is a 4-byte integer.

EIM_HANDLE_DOMAIN (1) The EIM domain name.

EIM_HANDLE_HOST (2) The host system for the EIM domain.

EIM_HANDLE_PORT (3) The port for the EIM connection. The returned field is a 4-byte integer.

EIM_HANDLE_SECPORT (4) Security type for this connection. The returned field is a 4-byte integer. Possible values are:

0 Non-SSL

1 Port uses SSL

EIM_HANDLE_MASTER_HOST (5)
If the EIM_HANDLE_HOST is a replica LDAP server, this value indicates the master LDAP server.

EIM_HANDLE_MASTER_PORT (6)
If the EIM_HANDLE_HOST is a replica LDAP server, this value indicates the port for the master LDAP server. The returned field is a 4-byte integer.

EIM_HANDLE_MASTER_SECPORT (7)
If the EIM_HANDLE_HOST is a replica LDAP server, this value indicates the security type for

the master LDAP server. The returned field is a 4-byte integer. Possible values are:

0 Non-SSL

1 Port uses SSL

lengthOfEimAttribute
(Input) The number of bytes the caller provides for the attribute information. The minimum size required is 16 bytes.

attribute
(Output) A pointer to the data to return. The EimAttribute structure contains information about the returned data. The API returns as much data as space has been provided. The EimAttribute structure follows:

```
typedef struct EimAttribute
{
    unsigned int bytesReturned; /* Number of bytes actually returned
                                   by the API                        */
    unsigned int bytesAvailable;/* Number of bytes of available data
                                   that could have been returned by
                                   the API                           */
    EimListData  attribute;     /* handle attribute                  */
} EimAttribute;
```

The EimListData structure follows:

```
typedef struct EimListData
{
    unsigned int length;        /* Length of data                    */
    unsigned int disp;          /* Displacement to data.  This byte
                                   offset is relative to the start of
                                   the parent structure, i.e. the
                                   structure containing this
                                   structure.                        */
} EimListData;
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimConnect” on page 215
- “eimConnectToMaster” on page 220
- “eimCreateHandle” on page 230
- “eimDestroyHandle” on page 239
- “eimSetAttribute” on page 383

Authorization

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_ATTR_INVAL (5)	Attribute name is not valid.
	EIMERR_ATTRIB_SIZE (53)	Length of EimAttribute is not valid.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
ENOTSUP	Attribute is not supported.	
	EIMERR_ATTR_NOTSUPP (6)	Attribute not supported.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example gets the distiguished name (DN) of the domain for the given EIM handle:

```
#include <eim.h>
#include <string.h>
#include <stdio.h>
 .
 .
 .
   int            rc;
   char           eimerr[200];
   EimRC        * err;
   EimHandle      handle;
   char         * data;
```

```
   char         * listData[1000];
   EimAttribute * list = (EimAttribute *) listData;

   /* Set up error structure.                  */
   memset(eimerr,0x00,200);
   err = (EimRC *)eimerr;
   err->memoryProvidedByCaller = 200;

.
.
.

   /* Get EIM domain name                      */
   if (0 != (rc = eimGetAttribute(&handle,
                                  EIM_HANDLE_DOMAIN,
                                  1000,
                                  list,
                                  err))) {
       char * errorString;
       if (NULL != (errorString = eimErr2String(err))) {
          printf("Get Attribute error = %d - %s\n", rc, errorString);
          free(errorString);
       } else {
          printf("Get Attribute error = %d - %s\n", rc, strerror(rc));
       }
   } else {
       data = (char * )list + list->attribute.disp;
       printf("Domain name = %s.\n", data);
   }
.
.
.
```

eimGetRegistryNameFromAlias

Purpose

Returns a list of registry names that match the search criteria that aliasType and aliasValue specify.

Format

```
#include <eim.h>


int eimGetRegistryNameFromAlias(EimHandle      * eim,
                                char           * aliasType,
                                char           * aliasValue,
                                unsigned int     lengthOfListData,
                                EimList        * listData,
                                EimRC          * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

aliasType
(Input) The type of alias for which to search. For a list of predefined alias types, see page 119.

aliasValue
(Input) The value of the alias to use for this search.

lengthOfListData
(Input) The number of bytes provided by the caller for the listData parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The minimum size required is 20 bytes.

listData
(Output) A pointer to the data to return. The EimList structure contains information about the returned data. The data returned is a linked list of EimRegistryName structures. The firstEntry is used to get to the first EimRegistryName structure in the linked list. The number of completed EimRegistryName structures is returned in entriesReturned. The bytesReturned variable has the number of bytes the API used for the returned entries. If the number of entries returned is less than the number of entries available, the returned data contains as many complete EimRegistryName structures as will fit. It can also contain a partial EimRegistryName structure.The EimList structure follows:

```
typedef struct EimList
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API                       */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API                          */
    unsigned int entriesReturned;   /* Number of entries actually
                                       returned by the API              */
    unsigned int entriesAvailable;  /* Number of entries available to be
                                       returned by the API              */
```

```
    unsigned int firstEntry;        /* Displacement to the first linked
                                       list entry. This byte offset is
                                       relative to the start of the
                                       EimList structure.              */
} EimList;
```

The EimRegistryName structure follows:

```
typedef struct EimRegistryName
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure         */
    EimListData name;               /* Name                            */
} EimRegistryName;
```

The EimListData structure follows:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                  */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure, i.e. the
                                       structure containing this
                                       structure.                      */
} EimListData;
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimChangeRegistryAlias” on page 207
- “eimListRegistryAliases” on page 325

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM identifiers administrator
- EIM registry *X* administrator
- EIM mapping lookup

The returned list contains only the information that the user has authority to access, meaning it could be empty.

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_EIMLIST_SIZE (16)	Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example gets the registry name from the specified alias:

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>


void printListResults(EimList * list);
void printListData(char * fieldName, void * entry, int offset);

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

```
    char           eimerr[200];
    EimRC        * err;
    EimHandle      handle;
    EimConnectInfo con;
    char         * ldapHost =
      "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";
    char           listData[1000];
    EimList      * list = (EimList * ) listData;

    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=admin";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Create handle with specified LDAP URL    */
    if (0 != (rc = eimCreateHandle(&handle,
                                   ldapHost,
                                   err))) {
         printf("Create handle error = %d\n", rc);
         return -1;
    }

    /* Connect with specified credentials       */
    if (0 != (rc = eimConnect(&handle,
                              con,
                              err))) {
         printf("Connect error = %d\n", rc);
         eimDestroyHandle(&handle, err);
         return -1;
    }

    /* Get all aliases for the registry         */
    if (0 != (rc = eimGetRegistryNameFromAlias(&handle,
                                               EIM_ALIASTYPE_DNS,
                                               "Clueless",
                                               1000,
                                               list,
                                               err)))
    {
        printf("List registry aliases error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    rc = eimDestroyHandle(&handle, err);

    return 0;
}

void printListResults(EimList * list)
{
    int i;
    EimRegistryName * entry;

    printf("___________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
```

```
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimRegistryName *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {

        /* Print out results */
        printListData("Registry Name",
                      entry,
                      offsetof(EimRegistryName, name));

        /* advance to next entry */
        entry = (EimRegistryName *)((char *)entry + entry->nextEntry);

    }
    printf("\n");
}

void printListData(char * fieldName, void * entry, int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}
```

eimGetTargetFromIdentifier

Purpose

Gets the target identity or identities for the specified registry that are associated with the specified EIM identifier.

Format

```
#include <eim.h>


int eimGetTargetFromIdentifier(EimHandle         * eim,
                               EimIdentifierInfo * idName,
                               char              * targetRegistryName,
                               char              * additionalInformation,
                               unsigned int        lengthOfListData,
                               EimList           * listData,
                               EimRC             * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

idName
(Input) A structure that contains the name of the identifier for this lookup operation. The layout of the EimIdentifierInfo structure follows:

```
enum EimIdType {
    EIM_UNIQUE_NAME,
    EIM_ENTRY_UUID,
    EIM_NAME
};


typedef struct EimIdentifierInfo
{
    union {
        char       * uniqueName;
        char       * entryUUID;
        char       * name;
    } id;
    enum EimIdType        idtype;
} EimIdentifierInfo;
```

`idtype`
The `idtype` in the EimIdentifierInfo structure indicates which identifier name has been provided. EIM_UNIQUE_NAME and EIM_ENTRY_UUID find at most one matching identifier. EIM_NAME results in an error if your EIM domain has more than one identifier containing the same name.

targetRegistryName
(Input) The target registry for this lookup operation. A null value for the string causes the service to use the system default local registry name from the instorage copy of the registry name.

additionalInfo
(Input) Additional information that is selection criteria for this operation. This can be a NULL string (for example, ″″). This filter data can contain the wildcard character, an asterisk (*). This field can be repeated and can contain more than one value.

lengthOfListData
(Input) The number of bytes provided by the caller for the listData parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The minimum size required is 20 bytes.

listData
(Output) A pointer to the data to return. The EimList structure contains information about the returned data. The data returned is a linked list of EimTargetIdentity structures. The firstEntry is used to get to the first EimTargetIdentity structure in the linked list. The number of completed EimTargetIdentity structures is returned in entriesReturned. The bytesReturned variable has the number of bytes the API used for the returned entries. If the number of entries returned is less than the number of entries available, the returned data contains as many complete EimTargetIdentity structures as will fit. It can also contain a partial EimTargetIdentity structure.The EimList structure follows:

```
typedef struct EimList
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API.                      */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API.                         */
    unsigned int entriesReturned;   /* Number of entries actually
                                       returned by the API.             */
    unsigned int entriesAvailable;  /* Number of entries available to be
                                       returned by the API.             */
    unsigned int firstEntry;        /* Displacement to the first linked
                                       list entry. This byte offset is
                                       relative to the start of the
                                       EimList structure.               */
} EimList;
```

The EimTargetIdentity structure follows:

```
typedef struct EimTargetIdentity
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure.         */
    EimListData userName;           /* User name                        */
} EimTargetIdentity;
```

The EimListData structure follows:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                   */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure, i.e. the
                                       structure containing this
                                       structure.                       */
} EimListData;
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- "eimGetTargetFromSource" on page 276

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM identifiers administrator
- EIM registry *X* administrator
- EIM mapping-lookup

The list returned contains only the information that the user has authority to access.

z/OS authorization

The calling application can be running in system key or supervisor state or one of the following:

- The RACF user ID of the caller's address space has READ authority to the BPX.SERVER profile in the FACILITY class
- The current RACF user ID has READ authority to the IRR.RGETINFO.EIM profile in the FACILITY class

The FACILITY class must be active and RACLISTed before unauthorized (problem program state and keys) will be granted the authority to use this SAF service.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
EBADNAME	Registry or identifier not found or insufficient access to EIM data.	
	EIMERR_IDNAME_AMBIGUOUS (20)	More than one EIM identifier was found that matches the requested Identifier name.
	EIMERR_NOIDENTIFIER (25)	EIM identifier not found or insufficient access to EIM data.
	EIMERR_NOREG (28)	EIM registry not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.

Return Value	Meaning	
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_EIMLIST_SIZE (16)	Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_IDNAME_TYPE_INVAL (52)	The EimIdType value is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
EMVSERR	An MVS environment or internal error has occurred.	
	EIMERR_ZOS_DATA_CONVERSION (6011)	Error occurred when converting data between code pages.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOSYS	EIM is not configured	
	EIMERR_NOTCONFIG (30)	(Only z/OS returns this value.) EIM environment is not configured. On z/OS, issue RACF commands to correct the configuration error and then try the request again.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use the eimConnect API or the eimConnectToMaster API and try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example gets the list of users in the target registry, MyRegistry, that is associated with the specified identifier:

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>


void printListResults(EimList * list);
```

```
void printListData(char * fieldName, void * entry, int offset);

int main(int argc, char *argv[])
{
    int               rc;
    char              eimerr[200];
    EimRC           * err;
    EimHandle         handle;
    EimConnectInfo    con;
    char            * ldapHost =
      "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";
    char              listData[4000];
    EimList         * list = (EimList * ) listData;
    EimIdentifierInfo x;

    /* Set up error structure.                  */
    Memset(eimerr, 0x00, 200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=admin";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Create handle with specified LDAP URL    */
    if (0 != (rc = eimCreateHandle(&handle,
                                   ldapHost,
                                   err))) {
        printf("Create handle error = %d\n", rc);
        return -1;
    }

    /* Connect with specified credentials       */
    if (0 != (rc = eimConnect(&handle,
                              con,
                              err))) {
        printf("Connect error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    /* Set up identifier information            */
    x.idtype = EIM_UNIQUE_NAME;
    x.id.uniqueName = "mjones";

    if (0 != (rc = eimGetTargetFromIdentifier(&handle,
                                              &x,
                                              "MyRegistry",
                                              NULL,
                                              4000,
                                              list,
                                              err)))
    {
        printf("Get Target from identifier error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    printListResults(list);

    /* Destroy the handle                       */
    rc = eimDestroyHandle(&handle, err);

    return 0;
}
```

```
void printListResults(EimList * list)
{
    int i;
    EimTargetIdentity * entry;

    printf("___________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimTargetIdentity *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("===============\n");
        printf("Entry %d.\n", i);

        /* Print out results */
        printListData("target user",
                      entry,
                      offsetof(EimTargetIdentity, userName));

        /* advance to next entry */
        entry = (EimTargetIdentity *)((char *)entry + entry->nextEntry);

    }
    printf("\n");

}

void printListData(char * fieldName, void * entry, int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}
```

eimGetTargetFromSource

Purpose

Gets the target identity or identities associated with the source identity (as defined by source registry name and source registry user). This is known as a mapping lookup operation -- from the known source information this API returns the user for this target registry.

The mapping lookup operation is done in the following order:

1. Check if both the source and target registries support mapping lookup operations. If not, no data is returned.
2. Specific source association to target association:
   - Check for source associations to EIM identifier(s) using the specified source registry user name and source registry. If none is found, proceeds to step 3.
   - Check for target associations to the EIM identifier(s) using the specified target registry. If none are found, proceeds to step 3.
   - If additional information is specified, check if any of the target identities have the same additional information. If not, proceeds to step 3.
   - Return the target identity(ies) for the specified target registry that have the same additional information that is specified.
3. Check if the domain supports policy associations. If not, no data is returned.
4. Check if the target registry supports policy associations. If not, no data is returned.
5. Certificate filter policy associations:
   - Check if the source registry is an X.509 registry. If not, proceeds to step 6.
   - Check if there is a certificate policy filter value that matches the source identity. If not, proceeds to step 6.
   - Check for certificate filter policy associations for the certificate filter policy value to the target registry. If none are found, proceeds to step 6.
   - If additional information is specified, check if any of the target identities have the same additional information. If not, proceeds to step 6.
   - Return the target identity(ies) for the specified target registry that have the same additional information that is specified.
6. Default registry policy associations:
   - Check for default registry policy associations for the source registry to the target registry. If none are found, proceeds to step 7.
   - If additional information is specified, check if any of the target identities have the same additional information. If not, proceeds to step 7.
   - Return the target identity(ies) for the specified target registry that have the same additional information that is specified.
7. Default domain policy associations:
   - Check for default domain policy associations to the target registry. If none are found, no data is returned.
   - If additional information is specified, check if any of the target identities have the same additional information. If not, no data is returned.
   - Return the target identity(ies) for the specified target registry that have the same additional information that is specified.

Format

```
#include <eim.h>


int eimGetTargetFromSource(EimHandle      * eim,
                           char           * sourceRegistryName,
                           char           * sourceRegistryUserName,
                           char           * targetRegistryName,
                           char           * additionalInformation,
                           unsigned int     lengthOfListData,
                           EimList        * listData,
                           EimRC          * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

sourceRegistryName
(Input) The source registry for this lookup operation. A null value for the string causes the service to use the system default local registry name from the instorage copy of the registry name.

sourceRegistryUserName
(Input) The source user name for this lookup operation. The registry user name should begin with a non-blank character.

targetRegistryName
(Input) The target registry for this lookup operation. A null value for the string causes the service to use the system default local registry name from the instorage copy of the registry name.

additionalInfo
(Input) Additional information to use as selection criteria for this operation. This can be NULL. This filter data can contain the wild card character, an asterisk (*).

lengthOfListData
(Input) The number of bytes provided by the caller for the listData parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The minimum size required is 20 bytes.

listData
(Output) A pointer to the data to return. The EimList structure contains information about the returned data. Entries are returned when the user is a member of the required EIM access group and the source and target registries exist. When the user is not a member of the required EIM access group or the target registry does not exist, the return value is zero and no entries are returned. The EimList structure follows:

```
typedef struct EimList
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API                       */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API                          */
    unsigned int entriesReturned;   /* Number of entries actually
                                       returned by the API              */
    unsigned int entriesAvailable;  /* Number of entries available to be
```

```
                                        returned by the API             */
    unsigned int firstEntry;           /* Displacement to the first linked
                                        list entry. This byte offset is
                                        relative to the start of the
                                        EimList structure.              */
} EimList;
```

The EimTargetIdentity structure follows:

```
typedef struct EimTargetIdentity
{
    unsigned int nextEntry;            /* Displacement to next entry.  This
                                        byte offset is relative to the
                                        start of this structure         */
    EimListData userName;              /* User name                       */
} EimTargetIdentity;
```

The EimListData structure follows:

```
typedef struct EimListData
{
    unsigned int length;               /* Length of data                  */
    unsigned int disp;                 /* Displacement to data.  This byte
                                        offset is relative to the start of
                                        the parent structure, i.e. the
                                        structure containing this
                                        structure.                      */
} EimListData;
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimGetTargetFromIdentifier” on page 270

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM identifiers administrator
- EIM mapping-lookup administrator
- EIM registry *X* administrator (for the source and target registries)

The list returned contains only the information that the user has authority to access.

z/OS authorization
The calling application can be running in system key or supervisor state or one of the following:

- The RACF user ID of the caller’s address space has READ authority to the BPX.SERVER profile in the FACILITY class

- The current RACF user ID has READ authority to the IRR.RGETINFO.EIM profile in the FACILITY class

The FACILITY class must be active and RACLISTed before unauthorized (problem program state and keys) will be granted the authority to use this SAF service.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful. If a target user ID is not returned in the listData and associations are defined between the source registry user ID and the target registry, ensure the user specified on the eimConnect or eimConnectToMaster is a member of the required EIM access group.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
EBADNAME	Source registry not found or insufficient access to EIM data.	
	EIMERR_NOREG (28)	EIM registry not found or the bind user specified on the EIM connect API is only a member of the target registry's EIM registry X administrator access group. If a target user ID is not returned in the listData and associations are defined between the source registry user ID and the target registry, ensure the user specified on the eimConnect or eimConnectToMaster is a member of the required EIM access group.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_EIMLIST_SIZE (16)	Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
EMVSERR	An MVS environment or internal error has occurred.	
	EIMERR_ZOS_DATA_CONVERSION (6011)	Error occurred when converting data between code pages.

Return Value	Meaning	
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOSYS	EIM is not configured	
	EIMERR_NOTCONFIG (30)	(Only z/OS returns this value.) EIM environment is not configured. On z/OS, issue RACF commands to correct the configuration error and then try the request again.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use the eimConnect API or eimConnectToMaster API and try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example gets the target identity that is associated with the source information:

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>


void printListResults(EimList * list);
void printListData(char * fieldName, void * entry, int offset);

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[200];
    EimRC       * err;
    EimHandle     handle;
    EimConnectInfo    con;
    char            * ldapHost =
      "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";
    char              listData[4000];
    EimList         * list = (EimList * ) listData;

    /* Set up error structure.                  */
    Memset(eimerr, 0x00, 200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=admin";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Create handle with specified LDAP URL    */
    if (0 != (rc = eimCreateHandle(&handle,
                                   ldapHost,
                                   err))) {
```

```
        printf("Create handle error = %d\n", rc);
        return -1;
    }

    /* Connect with specified credentials       */
    if (0 != (rc = eimConnect(&handle,
                              con,
                              err))) {
        printf("Connect error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    /* Get target identity                      */
    if (0 != (rc = eimGetTargetFromSource(&handle,
                                          "kerberosRegistry",
                                          "mjjones",
                                          "MyRegistry",
                                          NULL,
                                          4000,
                                          list,
                                          err)))
    {
        printf("Get Target from source error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    /* Destroy the handle                       */
    rc = eimDestroyHandle(&handle, err);

    return 0;
}

void printListResults(EimList * list)
{
    int i;
    EimTargetIdentity * entry;

    printf("__________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimTargetIdentity *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("===============\n");
        printf("Entry %d.\n", i);

        /* Print out results */
        printListData("target user",
                      entry,
                      offsetof(EimTargetIdentity, userName));

        /* advance to next entry */
        entry = (EimTargetIdentity *)((char *)entry + entry->nextEntry);

    }
    printf("\n");
```

```
}

void printListData(char * fieldName, void * entry, int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}
void printAssociationType(int type)
{
    switch(type)
    {
        case EIM_TARGET:
            printf("    Target Association.\n");
            break;
        case EIM_CERT_FILTER_POLICY:
            printf("    Certificate Filter Policy Association.\n");
            break;
        case EIM_DEFAULT_REG_POLICY:
            printf("    Default Registry Policy Association.\n");
            break;
        case EIM_DEFAULT_DOMAIN_POLICY:
            printf("    Default Domain Policy Association.\n");
            break;
        default:
            printf("ERROR - unknown association type.\n");
            break;
    }
}
```

eimGetVersion

Purpose

Returns the EIM version supported by the APIs for the specified host.

Format

```
#include <eim.h>
int eimGetVersion(EimHostInfo     * hostInfo,
                  enum EimVersion * version,
                  EimRC           * eimrc)
```

Parameters

eim

(Input) The EIM handle returned by a previous call to eimCreateHandle. A valid connection is required for this function.

hostInfo

(Input) The structure that contains the EIM host information for which to return the EIM version supported by the EIM APIs.

For EIM_HANDLE (0) host type, this field must contain an EIM handle returned by a previous call to eimCreateHandle and eimConnect.

For EIM_LDAP_URL (1) host type, this field must contain a uniform resource locator (URL) that contains the EIM host information. A NULL value for the ldapURL field indicates that the ldap URL information set by the eimSetConfiguration API should be used. This URL has the following format:

```
ldap://host:port/dn
```

or

```
ldaps://host:port
```

Where:

host:port
The name of the host on which the EIM domain controller is running with an optional port number.

dn The distinguished name of the domain to work with (optional).

ldaps Indicates that this host/port combination uses SSL and TLS.

The structure layout follows:

```
enum EimHostInfoType {
     EIM_HANDLE,
     EIM_LDAP_URL
};
typedef struct EimHostInfo {
     enum EimHostInfoType hostType;
     union {
       EimHandle * eim;
       char * ldapURL;
     } hostInfo;
} EimHostInfo;
```

version
(Output) The EIM version supported by the EIM APIs for the specified host. Possible values are:

EIM_VERSION_0 (0)
EIM is not supported on the specified host.

EIM_VERSION_1 (1)
EIM version 1 is supported by the EIM APIs for the specified host. This host will support EIM functionality provided with the first version of the EIM APIs .

EIM_VERSION_2 (2)
EIM version 2 is supported by the EIM APIs for the specified host. This host will support EIM functionality provided with the second version of the EIM APIs, which includes support for policy associations.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

None.

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.

Return Value	Meaning	
EINVAL	Input parameter was not valid.	
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_URL_NOHOST (47)	URL does not have a host.
	EIMERR_URL_NOTLDAP (49)	URL does not begin with ldap.
	EIMERR_TYPE_INVAL (69)	The specified type is not valid.
EMVSSAFXTRERR	SAF/RACF EXTRACT error.	
	EIMERR_ZOS_USER_XTR (6002)	RACROUTE REQUEST=EXTRACT error retrieving EIM configuration information from the caller's USER profile.
	EIMERR_ZOS_XTR_EIM(6003)	RACROUTE REQUEST=EXTRACT error retrieving EIM information from a RACF profile.
	EIMERR_ZOS_XTR_PROXY (6005)	RACROUTE REQUEST=EXTRACT error retrieving PROXY information from a RACF profile.
EMVSSAF2ERR	SAF/RACF error.	
	EIMERR_ZOS_XTR_DOMAINDN (6004)	EIM domain distinguished name is missing.
	EIMERR_ZOS_XTR_LDAPHOST (6006)	PROXY LDAP host is missing.
	EIMERR_ZOS_XTR_BINDDN (6007)	PROXY bind distinguished name is missing.
	EIMERR_ZOS_NO_ACEE (6010)	No task or address space ACEE found.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.
	EIMERR_LDAP_SCHEMA_NOT_FOUND (71)	Unable to find LDAP schema.

eimListAccess

Purpose

Lists the users that have the specified EIM access type.

Format

```
#include <eim.h>


int eimListAccess(EimHandle           * eim,
                  enum EimAccessType    accessType,
                  char                * registryName,
                  unsigned int          lengthOfListData,
                  EimList             * listData,
                  EimRC               * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

accessType
(Input) The type of access to list. Valid values are:

EIM_ACCESS_ADMIN (0) Administrative authority to the entire EIM domain.

EIM_ACCESS_REG_ADMIN (1)
Administrative authority to all registries in the EIM domain.

EIM_ACCESS_REGISTRY (2) Administrative authority to the registry specified in the *registryName* parameter.

EIM_ACCESS_IDENTIFIER_ADMIN (3)
Administrative authority to all of the identifiers in the EIM domain.

EIM_ACCESS_MAPPING_LOOKUP (4)
Authority to perform mapping lookup operations.

registryName
(Input) The name of the EIM registry for which to list access. Registry names are case-independent (meaning, not case-sensitive). If eimAccessType is anything other than EIM_ACCESS_REGISTRY, this parameter must be null.

The following special characters are not allowed in registry names:

```
, = + < > # ; \ *
```

lengthOfListData
(Input) The number of bytes the caller provides for the *listData* parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The minimum size required is 20 bytes.

listData
(Output) A pointer to the data to return.

The EimList structure contains information about the returned data. The data returned is a linked list of EimAccess structures. The firstEntry is used to get to the first EimAccess structures in the linked list. The number of completed EimAccess structures is returned in entriesReturned. The bytesReturned variable has the number of bytes the API used for the returned entries. If the number of entries returned is less than the number of entries available, the returned data contains as many complete EimAccess structures as will fit. It can also contain a partial EimAccess structures.The EimList structure follows:

```
typedef struct EimList
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API                       */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API                          */
    unsigned int entriesReturned;   /* Number of entries actually
                                       returned by the API              */
    unsigned int entriesAvailable;  /* Number of entries available to be
                                       returned by the API              */
    unsigned int firstEntry;        /* Displacement to the first linked
                                       list entry. This byte offset is
                                       relative to the start of the
                                       EimList structure.               */
} EimList;
```

The EimAccess structure follows:

```
typedef struct EimAccess
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure          */
    EimListData user;               /* User with access. This data will
                                       be in the format of the DN for
                                       for access id.                   */
} EimAccess;
```

The EimListData structure follows:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                   */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure, i.e. the
                                       structure containing this
                                       structure.                       */
} EimListData;
```

eimrc

(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddAccess” on page 166
- “eimListUserAccess” on page 344
- “eimQueryAccess” on page 351
- “eimRemoveAccess” on page 355

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

The list returned contains only the information that the user has authority to access.

z/OS authorization

No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_EIMLIST_SIZE (16)	Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_REG_MUST_BE_NULL (55)	Registry name must be NULL when access type is not EIM_ACCESS_REGISTRY.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.

Return Value	Meaning	
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example lists all users with access to the EIM Adminstrator access group:

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>


void printListResults(EimList * list);
void printListData(char * fieldName, void * entry, int offset);

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[200];
    EimRC       * err;
    EimHandle     handle;
    EimConnectInfo con;
    char         * ldapHost =
       "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";
    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=admin";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Create handle with specified LDAP URL    */
    if (0 != (rc = eimCreateHandle(&handle,
                                   ldapHost,
                                   err))) {
         printf("Create handle error = %d\n", rc);
         return -1;
    }

    /* Connect with specified credentials       */
    if (0 != (rc = eimConnect(&handle,
                              con,
                              err))) {
         printf("Connect error = %d\n", rc);
         eimDestroyHandle(&handle, err);
         return -1;
    }

    /* List all users with this access          */
    if (0 != (rc = eimListAccess(&handle,
                                 EIM_ACCESS_ADMIN,
                                 NULL,
                                 1000,
```

```
                                    list,
                                    err)))
    {
        printf("List access error = %d\n", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    /* Destroy the handle                       */
    rc = eimDestroyHandle(&handle, err);

    return 0;
}

void printListResults(EimList * list)
{
    int i;
    EimAccess * entry;

    printf("___________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimAccess *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("===============\n");
        printf("Entry %d.\n", i);

        /* Print out results */
        printListData("Access user",
                      entry,
                      offsetof(EimAccess, user));

        /* advance to next entry */
        entry = (EimAccess *)((char *)entry + entry->nextEntry);

    }
    printf("\n");

}
void printListData(char * fieldName, void * entry, int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");
```

eimListAssociations

Purpose

Returns a list of associations for a given EIM identifier. You can use this to find all of the associated identities for an individual in the enterprise. Note that this does not return policy associations.

Format

```
#include <eim.h>


int eimListAssociations(EimHandle               * eim,
                        enum EimAssociationType   associationType,
                        EimIdentifierInfo       * idName,
                        unsigned int              lengthOfListData,
                        EimList                 * listData,
                        EimRC                   * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

associationType
(Input) The type of association to list. Valid values are:

EIM_ALL_ASSOC (0)	List all associations.
EIM_TARGET (1)	List target associations.
EIM_SOURCE (2)	List source associations.
EIM_SOURCE_AND_TARGET (3)	List both source and target associations.
EIM_ADMIN (4)	List administrative associations.

idName
(Input) A structure that contains the identifier name indicating the associations to list. The layout of the EimIdentifierInfo structure follows:

```
enum EimIdType {
    EIM_UNIQUE_NAME,
    EIM_ENTRY_UUID,
    EIM_NAME
};


typedef struct EimIdentifierInfo
{
    union {
        char       * uniqueName;
        char       * entryUUID;
        char       * name;
    } id;
    enum EimIdType        idtype;
} EimIdentifierInfo;
```

`idtype`
The `idtype` in the EimIdentifierInfo structure indicates which identifier name has been provided. EIM_UNIQUE_NAME finds at most one matching

identifier. EIM_NAME results in an error if your EIM domain has more than one identifier containing the same name.

lengthOfListData
(Input) The number of bytes the caller provides for the *listData* parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. Minimum size required is 20 bytes.

listData
(Output) A pointer to the EimList structure.

The EimList structure contains information about the returned data. The data returned is a linked list of EimAssociation structures. The firstEntry field in the EimList structure is used to get to the first EimAssociation structure in the linked list. The number of completed EimAssociation structures is returned in entriesReturned. The bytesReturned variable has the number of bytes the API used for the returned entries. If the number of entries returned is less than the number of entries available, the returned data contains as many complete EimAssociation structures as will fit. It can also contain a partial EimAssociation structure.The EimList structure follows:

```
typedef struct EimList
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API                       */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API                          */
    unsigned int entriesReturned;   /* Number of entries actually
                                       returned by the API              */
    unsigned int entriesAvailable;  /* Number of entries available to be
                                       returned by the API              */
    unsigned int firstEntry;        /* Displacement to the first linked
                                       list entry. This byte offset is
                                       relative to the start of the
                                       EimList structure.               */
} EimList;
```

The EimAssociation structure follows:

```
typedef struct EimAssociation
   {
       unsigned int nextEntry;         /* Displacement to next entry.  This
                                          byte offset is relative to the
                                          start of this structure          */
       enum EimAssociationType associationType; /* Type of association   */
       EimListData registryType;       /* Registry type                    */
       EimListData registryName;       /* Registry name                    */
       EimListData registryUserName;   /* Registry user name               */
   } EimAssociation;
```

The EimListData structure follows:

```
 typedef struct EimListData
{
    unsigned int length;            /* Length of data                   */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure, i.e. the
                                       structure containing this
                                       structure.                       */
} EimListData;
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddAssociation” on page 174
- “eimGetAssociatedIdentifiers” on page 254
- “eimRemoveAssociation” on page 359

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM identifiers administrator
- EIM registry *X* administrator
- EIM mapping lookup

The list returned contains only the information that the user has authority to access.

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning
0	Request was successful.
EACCES	Access denied. Not enough permissions to access data.
EBADDATA	eimrc is not valid.
EBADNAME	Identifier name is not valid. **EIMERR_IDNAME_AMBIGUOUS (20)** More than one EIM identifier was found that matches the requested identifier name. **EIMERR_NOIDENTIFIER (25)** EIM identifier not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object. **EIMERR_NOLOCK (26)** (z/OS does not return this value.) Unable to allocate internal system object.

Return Value	Meaning
ECONVERT	Data conversion error.
	EIMERR_DATA_CONVERSION (13) (z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.
	EIMERR_ASSOC_TYPE_INVAL (4) Association type is not valid.
	EIMERR_EIMLIST_SIZE (16) Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_HANDLE_INVAL (17) EimHandle is not valid.
	EIMERR_IDNAME_TYPE_INVAL (52) The EimIdType value is not valid.
	EIMERR_PARM_REQ (34) Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35) (z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41) Unexpected error accessing parameter.
ENOMEM	Unable to allocate required space.
	EIMERR_NOMEM (27) No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.
	EIMERR_NOT_CONN (31) Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EUNKNOWN	Unexpected exception.
	EIMERR_LDAP_ERR (23) Unexpected LDAP error.
	EIMERR_UNEXP_OBJ_VIOLATION (56) Unexpected object violation.
	EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Example

The following example lists the associations for an identifier:

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>


void printListResults(EimList * list);
void printAssociationType(int type);
void printListData(char * fieldName, void * entry, int offset);

int main(int argc, char *argv[])
{
    int               rc;
    char              eimerr[200];
    EimRC           * err;
    EimHandle         handle;
    EimConnectInfo    con;
    char            * ldapHost =
      "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";
```

```
    char              listData[4000];
    EimList         * list = (EimList * ) listData;
    EimIdentifierInfo x;

    /* Set up error structure.                  */
    Memset(eimerr, 0x00, 200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=admin";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Create handle with specified LDAP URL    */
    if (0 != (rc = eimCreateHandle(&handle,
                                   ldapHost,
                                   err))) {
        printf("Create handle error = %d\n", rc);
        return -1;
    }

    /* Connect with specified credentials       */
    if (0 != (rc = eimConnect(&handle,
                              con,
                              err))) {
        printf("Connect error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    /* Set up identifier information            */
    x.idtype = EIM_UNIQUE_NAME;
    x.id.uniqueName = "mjones";

    /* Get associations for this identifier     */
    if (0 != (rc = eimListAssociations(&handle,
                                       EIM_ALL_ASSOC,
                                       &x,
                                       4000,
                                       list,
                                       err)))
    {
        printf("List Association error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    /* Destroy the handle                       */
    rc = eimDestroyHandle(&handle, err);

    return 0;
}

void printListResults(EimList * list)
{
    int i;
    EimAssociation * entry;

    printf("___________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
```

```
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimAssociation *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("===============\n");
        printf("Entry %d.\n", i);

        /* Association type */
        printAssociationType(entry->associationType);

        /* Print out results */
        printListData("Registry Type",
                      entry,
                      offsetof(EimAssociation, registryType));
        printListData("Registry Name",
                      entry,
                      offsetof(EimAssociation, registryName));
        printListData("Registry User Name",
                      entry,
                      offsetof(EimAssociation, registryUserName));

        /* advance to next entry */
        entry = (EimAssociation *)((char *)entry + entry->nextEntry);

    }
    printf("\n");

}

void printAssociationType(int type)
{
    switch(type)
    {
        case EIM_TARGET:
            printf("    Target Association.\n");
            break;
        case EIM_SOURCE:
            printf("    Source Association.\n");
            break;
        case EIM_ADMIN:
            printf("    Admin Association.\n");
            break;
        default:
            printf("ERROR - unknown association type.\n");
            break;
    }
}

void printListData(char * fieldName, void * entry, int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
```

```
    else
        printf("Not found.\n");

}
```

eimListDomains

Purpose

Lists information for a single EIM domain or for all EIM domains that are stored on an LDAP server. To list a single domain, the *ldapURL* parameter must contain the distinguished name of the EIM domain.

To list all domains stored on an LDAP server, omit the distinguished name of the EIM domain from the *ldapURL* parameter.

Format

```
#include <eim.h>


int eimListDomains(char           * ldapURL,
                   EimConnectInfo   connectInfo,
                   unsigned int     lengthOfListData,
                   EimList        * listData,
                   EimRC          * eimrc)
```

Parameters

ldapURL
(Input) A uniform resource locator (URL) that contains the EIM host information. This parameter is required. This URL has the following format:

```
ldap://host:port/dn
```

or

```
ldaps://host:port/dn
```

host:port
Name of the host on which the EIM domain controller is running. (The port number is optional. If not specified, the default LDAP or LDAPS port will be used.)

dn Distinguished name of the domain to list. If you do not specify DN, then eimListDomains returns all domains stored on an LDAP server.

Examples:

```
ldap://systemx:389/ibm-eimDomainName=myEimDomain,o=myCompany,c=us
ldaps://systemy:636/ibm-eimDomainName=thisEimDomain,o=myCompany,c=us
```

Note: In contrast with `ldap`, `ldaps` indicates that this host and port combination uses SSL and TLS.

connectInfo
(Input) Connect information. This parameter provides the information required to bind to LDAP. If the system is configured to connect to a secure port, EimSSLInfo is required.

For the EIM_SIMPLE connect type, the `creds` field should contain the EimSimpleConnectInfo structure with a binddn and password.

EimPasswordProtect determines the level of password protection on the LDAP bind.

EIM_PROTECT_NO (0) The clear-text password is sent on the bind.

EIM_PROTECT_CRAM_MD5 (1)
The protected password is sent on the bind.

The server side must support cram-md5 protocol to send the protected password.

EIM_PROTECT_CRAM_MD5_OPTIONAL (2)
The protected password is sent on the bind if the cram-md5 protocol is supported. Otherwise, the clear-text password is sent.

For EIM_KERBEROS, the default logon credentials are used. The `kerberos_creds` field must be NULL.

For EIM_CLIENT_AUTHENTICATION, the `creds` field is ignored. The `ssl` field must point to a valid EimSSLInfo structure. The `keyring` field is required in the EimSSLInfo structure. It can be the name of a System SSL key database file or a RACF keyring name. The `keyring_pw` field is required when the keyring is the name of a System SSL key database field. The `certificateLabel` field is optional. If it is NULL the default certificate in the keyring is used.

The structure layouts follow:

```
enum EimPasswordProtect {
    EIM_PROTECT_NO,
    EIM_PROTECT_CRAM_MD5,
    EIM_PROTECT_CRAM_MD5_OPTIONAL
};

enum EimConnectType {
    EIM_SIMPLE,
    EIM_KERBEROS,
    EIM_CLIENT_AUTHENTICATION
};


typedef struct EimSimpleConnectInfo
{
     enum EimPasswordProtect protect;
     char * bindDn;
     char * bindPw;
} EimSimpleConnectInfo;

typedef struct EimSSLInfo
{
     char * keyring;
     char * keyring_pw;
     char * certificateLabel;
} EimSSLInfo;


typedef struct EimConnectInfo
{
     enum EimConnectType type;
     union {
         gss_cred_id_t * kerberos;
         EimSimpleConnectInfo simpleCreds;
     } creds;
   EimSSLInfo * ssl;
} EimConnectInfo;
```

lengthOfListData
(Input) The number of bytes the caller provides for the list of domains. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The API returns the number of bytes available for the entire list and as much data as space has been provided. Minimum size required is 20 bytes.

listData

(Output) A pointer to the data to return. The EimList structure contains information about the returned data. The data returned is a linked list of EimDomain structures. The firstEntry field in the EimList Structure is used to get to the first EimDomain structure in the linked list. The number of completed EimDomain structures is returned in entriesReturned. The bytesReturned variable has the number of bytes the API used for the returned entries. If the number of entries returned is less than the number of entries available, the returned data contains as many complete EimDomain structures as will fit. It can also contain a partial EimDomain structure.The EimList structure follows:

```
typedef struct EimList
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API                       */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API                          */
    unsigned int entriesReturned;   /* Number of entries actually
                                       returned by the API              */
    unsigned int entriesAvailable;  /* Number of entries available to be
                                       returned by the API              */
    unsigned int firstEntry;        /* Displacement to the first linked
                                       list entry. This byte offset is
                                       relative to the start of the
                                       EimList structure.               */
} EimList;
```

The EimDomain structure follows:

```
typedef struct EimDomain
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure          */
    EimListData name;               /* Domain name                      */
    EimListData DN;                 /* Distinguished name for the domain
              */
    EimListData description;        /* Description                      */
    enum EimStatus  policyAssociations;  /* Policy associations
                                            attribute         @01A*/
} EimDomain;
```

The EimListData structure follows:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                   */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure, i.e. the
                                       structure containing this
                                       structure.                        */
} EimListData;
```

eimrc

(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimChangeDomain” on page 194

- “eimCreateDomain” on page 225
- “eimDeleteDomain” on page 234

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

The list returned contains only the information that the user has authority to access.

z/OS authorization

No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning
0	Request was successful.
EACCES	Access denied. Not enough permissions to access data.
EBADDATA	eimrc is not valid.
EBADNAME	EIM domain not found or insufficient access to EIM data. **EIMERR_NODOMAIN (24)** EIM domain not found or insufficient access to EIM data.
ECONVERT	Data conversion error. **EIMERR_DATA_CONVERSION (13)** (z/OS does not return this value.) Error occurred when converting data between code pages.

Return Value	Meaning	
EINVAL	Input parameter was not valid.	
	EIMERR_CONN_INVAL (54)	Connection type is not valid.
	EIMERR_EIMLIST_SIZE (16)	Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_NOT_SECURE (32)	The system is not configured to connect to a secure port. Connection type of EIM_CLIENT_AUTHENTICATION is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PROTECT_INVAL (22)	The protect parameter in EimSimpleConnectInfo is not valid.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
	EIMERR_SSL_REQ (42)	The system is configured to connect to a secure port. EimSSLInfo is required.
	EIMERR_URL_NOHOST (47)	URL does not have a host.
	EIMERR_URL_NOTLDAP (49)	URL does not begin with ldap.
	EIMERR_CREDS_MUST_BE_NULL (58)	The connectInfo parameter of the EIM API does not have a NULL value for the creds field in the EimConnectInfo structure.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTSUP	Connection type is not supported.	
	EIMERR_CONN_NOTSUPP (12)	Connection type is not supported.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example lists the information for the specified EIM domain:

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void printListResults(EimList * list);
void printListData(char * fieldName, void * entry, int offset);

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[200];
    EimRC       * err;
    char          listData[1000];
    EimList     * list = (EimList * ) listData;
```

```
    char        * errstr;

    char * ldapURL = "ldap://localhost:389";

    EimConnectInfo con;

    /* Set up connection information            */
    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=administrator";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    /* Get info for specified domain            */
    if (0 != (rc = eimListDomains(ldapURL,
                                  con,
                                  1000,
                                  list,
                                  err)))
    {
       printf("List domain error = %d - %s\n", rc, errstr=eimErr2String(err));
       if (NULL != errstr) free(errstr);
    return -1;
    }

    /* Print the results                        */
    printListResults(list);
    return 0;
}

void printListResults(EimList * list)
{
    int              i;
    EimDomain      * entry;
    EimListData    * listData;
    char           * data;
    int              dataLength;
    enum EimStatus * status;

    printf("___________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimDomain *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("===============\n");
        printf("Entry %d.\n", i);

        /* Print out results */
        printListData("Domain Name",
                      entry,
                      offsetof(EimDomain, name));
        printListData("Domain DN",
                      entry,
                      offsetof(EimDomain, dn));
        printListData("description",
                      entry,
```

```
                        offsetof(EimDomain, description));
            printf("     Policy Associations are ");

            status = (enum EimStatus *)((char *)entry +
                                        offsetof(EimDomain, policyAssociations));
            if ( *status == EIM_STATUS_ENABLED) {
               printf("Enabled\n");
            } else {
               printf("Disabled\n");
            }

            /* advance to next entry */
            entry = (EimDomain *)((char *)entry + entry->nextEntry);

        }
        printf("\n");

    }

    void printListData(char * fieldName, void * entry, int offset)
    {
        EimListData * listData;
        char * data;
        int dataLength;

        printf("     %s = ",fieldName);
        /* Address the EimListData object */
        listData = (EimListData *)((char *)entry + offset);

        /* Print out results */
        data = (char *)entry + listData->disp;
        dataLength = listData->length;

        if (dataLength > 0)
            printf("%.*s\n",dataLength, data);
        else
            printf("Not found.\n");

    }
```

eimListIdentifiers

Purpose

Returns a list of identifiers in the EIM domain. idName can be used to filter the results returned.

Format

```
#include <eim.h>


int eimListIdentifiers(EimHandle         * eim,
                       EimIdentifierInfo * idName,
                       unsigned int        lengthOfListData,
                       EimList           * listData,
                       EimRC             * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

idName
(Input) A structure that contains the name for this identifier. This parameter can be NULL; in this case the API returns all identifiers in the domain. The layout of the EimIdentifierInfo structure follows:

```
enum EimIdType {
    EIM_UNIQUE_NAME,
    EIM_ENTRY_UUID,
    EIM_NAME
};


typedef struct EimIdentifierInfo
{
    union {
        char       * uniqueName;
        char       * entryUUID;
        char       * name;
    } id;
    enum EimIdType        idtype;
} EimIdentifierInfo;
```

`idtype`
The `idtype` in the EimIdentifierInfo structure indicates which identifier name has been provided. There is no guarantee that `name` will find a unique identifier. Therefore, using `name` can result in the return of multiple identifiers. The id values `uniqueName`, `entryUUID` and `name` can contain the wild card character, an asterisk (*).

lengthOfListData
(Input) The number of bytes the caller provides for the *listData* parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The minimum size required is 20 bytes.

listData
(Output) A pointer to the EimList structure. The EimList structure contains information about the returned data. The data returned is a linked list of

EimIdentifier structures. The firstEntry field in the EimList structure is used to get to the first EimIdentifier structure in the linked list. The number of completed EimIdentifier structures is returned in entriesReturned. The bytesReturned variable has the number of bytes the API used for the returned entries. If the number of entries returned is less than the number of entries available, the returned data contains as many complete EimIdentifier structures as will fit. It can also contain a partial EimIdentifier structure.The EimList structure follows:

```
typedef struct EimList
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API                       */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API                          */
    unsigned int entriesReturned;   /* Number of entries actually
                                       returned by the API              */
    unsigned int entriesAvailable;  /* Number of entries available to be
                                       returned by the API              */
    unsigned int firstEntry;        /* Displacement to the first linked
                                       list entry. This byte offset is
                                       relative to the start of the
                                       EimList structure.               */
} EimList;
```

The EimIdentifier structure follows:

```
typedef struct EimIdentifier
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure          */
    EimListData uniquename;         /* Unique name                      */
    EimListData description;        /* Description                      */
    EimListData entryUUID;          /* UUID                             */
    EimSubList  names;              /* EimIdentifierName sublist        */
    EimSubList  additionalInfo;     /* EimAddlInfo sublist              */
} EimIdentifier;
```

Identifiers might have defined several name attributes as well as several additional information attributes. In the EimIdentifier structure, the names EimSubList gives addressability to a linked list of EimIdentifierName structures. The EimIdentifierName follows:

```
typedef struct EimIdentifierName
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure          */
    EimListData name;               /* Name                             */
} EimIdentifierName;
```

The additionalInfo EimSubList gives addressability to a linked list of EimAddlInfo structures. The EimAddlInfo structure follows:

```
typedef struct EimAddlInfo
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure          */
    EimListData addlInfo;           /* Additional info                  */
} EimAddlInfo;
```

The EimSubList structure follows:

```
typedef struct EimSubList
{
    unsigned int listNum;          /* Number of entries in the list  */
    unsigned int disp;             /* Displacement to sublist. This
                                      byte offset is relative to the
                                      start of the parent structure, i.e.
                                      the structure containing this
                                      structure.                     */
} EimSubList;
```

The EimListData structure follows:

```
typedef struct EimListData
{
    unsigned int length;           /* Length of data                 */
    unsigned int disp;             /* Displacement to data.  This byte
                                      offset is relative to the start of
                                      the parent structure, i.e. the
                                      structure containing this
                                      structure.                     */
} EimListData;
```

eimrc

(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddIdentifier” on page 179
- “eimChangeIdentifier” on page 199
- “eimGetAssociatedIdentifiers” on page 254
- “eimRemoveIdentifier” on page 364

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM identifiers administrator
- EIM registry *X* administrator
- EIM mapping lookup

The list returned contains only the information that the user has authority to access.

z/OS authorization

No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
EBADNAME	Identifier name is not valid.	
	EIMERR_NOIDENTIFIER (25)	EIM identifier not found.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_EIMLIST_SIZE (16)	Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_IDNAME_TYPE_INVAL (52)	The EimIdType value is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates listing all EIM identifiers:

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>


void printListResults(EimList * list);
void printSubListData(char * fieldName, void * entry, int offset);
void printListData(char * fieldName, void * entry, int offset);

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

```
    int            rc;
    char           eimerr[200];
    EimRC        * err;
    EimHandle      handle;
    EimConnectInfo con;
    char         * ldapHost =
     "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";
    char           listData[4000];
    EimList      * list = (EimList * ) listData;

    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=admin";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Create handle with specified LDAP URL    */
    if (0 != (rc = eimCreateHandle(&handle,
                                   ldapHost,
                                   err))) {
        printf("Create handle error = %d\n", rc);
        return -1;
    }

    /* Connect with specified credentials       */
    if (0 != (rc = eimConnect(&handle,
                              con,
                              err))) {
        printf("Connect error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    /* Get all identifiers                      */
    if (0 != (rc = eimListIdentifiers(&handle,
                                      NULL,
                                      4000,
                                      list,
                                      err)))
    {
        printf("List identifiers error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    /* Destroy the handle                       */
    rc = eimDestroyHandle(&handle, err);

    return 0;
}

void printListResults(EimList * list)
{
    int i;
    EimIdentifier * entry;

    printf("___________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
```

```
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimIdentifier *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("===============\n");
        printf("Entry %d.\n", i);

        /* Print out results */
        printListData("Unique name",
                      entry,
                      offsetof(EimIdentifier, uniquename));
        printListData("description",
                      entry,
                      offsetof(EimIdentifier, description));
        printListData("entryUUID",
                      entry,
                      offsetof(EimIdentifier, entryUUID));
        printSubListData("Names",
                         entry,
                         offsetof(EimIdentifier, names));
        printSubListData("Additional Info",
                         entry,
                         offsetof(EimIdentifier, additionalInfo));

        /* advance to next entry */
        entry = (EimIdentifier *)((char *)entry + entry->nextEntry);

    }
    printf("\n");

}

void printSubListData(char * fieldName, void * entry, int offset)
{
    int i;
    EimSubList * subList;
    EimAddlInfo * subentry;

    /* Address the EimSubList object */
    subList = (EimSubList *)((char *)entry + offset);

    if (subList->listNum > 0)
    {
        subentry = (EimAddlInfo *)((char *)entry + subList->disp);
        for (i = 0; i < subList->listNum; i++)
        {
            /* Print out results */
            printListData(fieldName, subentry,
                          offsetof(EimAddlInfo, addlInfo));

            /* advance to next entry */
            subentry = (EimAddlInfo *)((char *)subentry +
                                       subentry->nextEntry);
        }
    }
}

void printListData(char * fieldName, void * entry, int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;
```

```
    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}
```

eimListPolicyFilters

Purpose

Lists policy filters for the domain.

Format

```
#include <eim.h>
int eimListPolicyFilters(EimHandle                *  eim,
                         enum EimPolicyFilterType    filterType,
                         char                     *  registryName,
                         unsigned int                lengthOfListData,
                         EimList                  *  listData,
                         EimRC                    *  eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

filterType
(Input) The type of policy filters to be listed. Valid values are:

EIM_ALL_FILTERS (0)
List all policy filters.

EIM_CERTIFICATE_FILTER (1)
List certificate policy filters.

registryName
(Input) The name of the X509 registry for which you want to list policy filters. If NULL is specified, then policy filters for the entire domain are listed.

lengthOfListData
(Input) The number of bytes provided by the caller for the listData parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The minimum size is 20 bytes.

listData
(Output) A pointer to the EimList structure, which contains information about the returned data. The API returns as much data as space allows. The data returned is a linked list of EimPolicyFilter structures. The EimList structure follows:

```
typedef struct EimList
{
  unsigned int bytesReturned;   /* Number of bytes actually returned
                                by the API */
  unsigned int bytesAvailable;  /* Number of bytes of available data
                                that could have been returned by
                                the API */
  unsigned int entriesReturned; /* Number of entries actually
                                returned by the API */
  unsigned int entriesAvailable; /* Number of entries available to be
                                returned by the API */
  unsigned int firstEntry;      /* Displacement to the first linked
                                list entry. This byte offset is
                                relative to the start of the
                                EimList structure. */
} EimList;
```

The EimPolicyFilter structure follows:

```
typedef struct EimPolicyFilter
{
  unsigned int nextEntry;        /* Displacement to next entry. This
                                 byte offset is relative to the
                                 start of this structure */
  enum EimPolicyFilterType type; /* Type of policy filter. */
  EimListData sourceRegistry;    /* Source registry name the policy
                                 filter is defined for. */
  EimListData filterValue;       /* Policy filter value. */
} EimPolicyFilter;
```

The EimListData structure follows:

```
typedef struct EimListData
{
   unsigned int length; /* Length of data */
   unsigned int disp;   /* Displacement to data. This byte
                         offset is relative to the start of
                         the parent structure; that is, the
                         structure containing this
                         structure. */
} EimListData;
```

eimrc

(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddPolicyFilter” on page 187
- “eimRemovePolicyFilter” on page 371

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM identifiers administrator
- EIM Mapping Lookup
- EIM registry authority

The list returned (which can be empty) contains only the information that the user has authority to access.

z/OS authorization

No special authority is needed.

Return Values

Return Value	Meaning
0	Request was successful.

Return Value	Meaning	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	(z/OS does not return this value.) Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBADNAME	Identifier name is not valid or insufficient access to EIM data.	
	EIMERR_NOREG (28)	EIM Registry not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_EIMLIST_SIZE (16)	Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified or configured EIM Domain controller does not support this API.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
	EIMERR_POLICY_FILTER_TYPE_INVAL (60)	Policy filter type is not valid.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.
	EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.

Example

The following example lists certificate policy filters for a registry.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
```

```
#include <string.h>

void printListResults(EimList * list);
void printListData(char * fieldName,
                   void * entry,
                   int    offset);

int main (int argc, char *argv[])
{
    int            rc;
    char           eimerr[200];
    EimRC        * err;
    EimHandle      handle;
    char           listData[1000];
    EimList      * list = (EimList *)listData;
    EimConnectInfo con;
    char         * ldapHost =
      "ldap://localhost:389/ibm-eimDomainName=MyDomain,o=MyCompany,c=us";


    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=administrator";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Create handle with specified LDAP URL    */
    if (0 != (rc = eimCreateHandle(&handle, ldapHost, err))) {
         printf("Create handle error = %d\n", rc);
         return -1;
    }

    /* Connect with specified credentials       */
    if (0 != (rc = eimConnect(&handle, con, err))) {
         printf("Connect error = %d\n", rc);
         eimDestroyHandle(&handle, err);
         return -1;
    }

    /* Get source registry policies */
    if (0 != (rc = eimListPolicyFilters(&handle,
                                        EIM_ALL_FILTERS,
                                        NULL,
                                        1000,
                                        list,
                                        err)))
    {
         printf("List EIM Policy Filters error = %d", rc);
         eimDestroyHandle(&handle, err);
         return -1;
    }

    /* Print the results */
    printListResults(list);

    /* Destroy the handle */
    rc = eimDestroyHandle(&handle, err);

    return 0;
}

void printListResults(EimList * list)
```

```
{
    int i;
    EimPolicyFilter * entry;

    printf("__________\n");
    printf(" bytesReturned = %d\n", list->bytesReturned);
    printf(" bytesAvailable = %d\n", list->bytesAvailable);
    printf(" entriesReturned = %d\n", list->entriesReturned);
    printf(" entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimPolicyFilter *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("===============\n");
        printf("Entry %d.\n", i);

        /* Print out results */
        printListData("Source Registry",
                      entry,
                      offsetof(EimPolicyFilter, sourceRegistry));
        printListData("Filter Value",
                      entry,
                      offsetof(EimPolicyFilter, filterValue));

        /* advance to next entry */
        entry = (EimPolicyFilter *)((char *)entry + entry->nextEntry);
    }

    printf("\n");
}

void printListData(char * fieldName,
                   void * entry,
                   int    offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("    %s = ",fieldName);

    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
         printf("%.*s\n",dataLength, data);
    else
         printf("Not found.\n");
}
```

eimListRegistries

Purpose

Lists the user registries participating in the EIM domain. You can use the *registryType*, *registryName* and *registryKind* parameters to filter the results returned.

Format

```
#include <eim.h>


int eimListRegistries(EimHandle           * eim,
                      char                * registryName,
                      char                * registryType,
                      enum EimRegistryKind  registryKind,
                      unsigned int          lengthOfListData,
                      EimList             * listData,
                      EimRC               * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

registryName
(Input) The name of the EIM registry to list. The name can contain the wild card character, an asterisk (*). This is used as a filter to determine which registries to return. This parameter can be NULL; in this case, no filtering is done by name. Registry names are case-independent (meaning, not case-sensitive).

registryType
(Input) A string form of an OID that represents the registry type and a user name normalization method. The normalization method is necessary because some registries are case-independent and others are case-dependent. EIM uses this information to make sure the appropriate search occurs for registry user names.

The predefined registry types that EIM provides include the following:

- EIM_REGTYPE_AIX
- EIM_REGTYPE_DOMINO_LONG
- EIM_REGTYPE_DOMINO_SHORT
- EIM_REGTYPE_KERBEROS_EX
- EIM_REGTYPE_KERBEROS_IG
- EIM_REGTYPE_LDAP
- EIM_REGTYPE_LINUX
- EIM_REGTYPE_NDS
- EIM_REGTYPE_OS400
- EIM_REGTYPE_POLICY_DIRECTOR
- EIM_REGTYPE_RACF
- EIM_REGTYPE_TIVOLI_ACCESS_MANAGER
- EIM_REGTYPE_WIN2K
- EIM_REGTYPE_WIN_DOMAIN_KERB_IG
- EIM_REGTYPE_WINDOWS_LOCAL_WS

- EIM_REGTYPE_X509

You can also create your own registry type. This parameter can also be NULL; in this case, the API returns all registry types.

registryKind
(Input) The kind of registry to list. Valid values are:

EIM_ALL_REGISTRIES (0) EIM returns both system and application registries.

EIM_SYSTEM_REGISTRY (1) EIM returns only system registries.

EIM_APPLICATION_REGISTRY (2)
EIM returns only application registries.

lengthOfListData
(Input) The number of bytes the caller provids for the *listData* parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The minimum size required is 20 bytes.

listData
(Output) A pointer to the data to return. The EimList structure contains information about the returned data. The data returned is a linked list of EimRegistry structures. The firstEntry field in the EimList structure is used to get to the first EimRegistry structure in the linked list. The number of completed EimRegistry structures is returned in entriesReturned. The bytesReturned variable has the number of bytes the API used for the returned entries. If the number of entries returned is less than the number of entries available, the returned data contains as many complete EimRegistry structures as will fit. It can also contain a partial EimRegistry structure.The EimList structure follows:

```
typedef struct EimList
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API                       */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API                          */
    unsigned int entriesReturned;   /* Number of entries actually
                                       returned by the API              */
    unsigned int entriesAvailable;  /* Number of entries available to be
                                       returned by the API              */
    unsigned int firstEntry;        /* Displacement to the first linked
                                       list entry. This byte offset is
                                       relative to the start of the
                                       EimList structure.               */
} EimList;
```

The EimRegistry structure follows:

```
typedef struct EimRegistry
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure          */
    enum EimRegistryKind  kind;     /* Kind of registry                 */
    EimListData name;               /* Registry name                    */
    EimListData type;               /* Registry type                    */
    EimListData description;        /* Description                      */
    EimListData entryUUID;          /* Entry UUID                       */
    EimListData URI;                /* URI                              */
    EimListData systemRegistryName; /* System registry name             */
    EimSubList  registryAlias;      /* EimRegistryAlias sublist         */
```

```
    enum EimStatus  mappingLookup;  /* Mapping lookup attribute      */
    enum EimStatus  policyAssociations; /* Policy associations
                                           attribute                  */
} EimRegistry;
```

Registries can have a number of defined aliases. In the EimRegistry structure, the registryAlias EimSubList gives addressability to the first EimRegistryAlias structure. The EimRegistryAlias structure follows:

```
typedef struct EimRegistryAlias
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure          */
    EimListData type;               /* Alias type                       */
    EimListData value;              /* Alias value                      */
} EimRegistryAlias;
```

The EimSubList structure follows:

```
typedef struct EimSubList
{
    unsigned int listNum;           /* Number of entries in the list  */
    unsigned int disp;              /* Displacement to sublist. This
                                       byte offset is relative to the
                                       start of the parent structure i.e.
                                       the structure containing this
                                       structure                        */
} EimSubList;
```

The EimListData structure follows:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                 */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure, i.e. the
                                       structure containing this
                                       structure.                       */
} EimListData;
```

eimrc

(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddApplicationRegistry” on page 170
- “eimAddSystemRegistry” on page 190
- “eimChangeRegistry” on page 203
- “eimRemoveRegistry” on page 374

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

- EIM registries administrator
- EIM identifiers administrator
- EIM registry *X* administrator
- EIM mapping lookup

The list returned (which can be empty) contains only the information that the user has authority to access.

z/OS authorization

No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_EIMLIST_SIZE (16)	Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_REGKIND_INVAL (38)	Requested registry kind is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example lists all registries found:

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

void printRegistryKind(int kind);
void printListResults(EimList * list);
void printListData(char * fieldName, void * entry, int offset);
void printAliasSubList(void * entry, int offset);

int main (int argc, char *argv[])
{
    int            rc;
    char           eimerr[200];
    EimRC        * err;
    EimHandle      handle;
    EimConnectInfo con;
    char         * errstr;
    char         * ldapHost =
       "ldap://localhost:389/ibm-eimdomainname=MyDomain,o=MyCompany,c=US";


    char          listData[16000];
    EimList     * list = (EimList * ) listData;

    /* Set up error structure.                  */
    memset(eimerr, 0x00, 200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=administrator";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Create handle with specified LDAP URL    */
    if (0 != (rc = eimCreateHandle(&handle,
                                   ldapHost,
                                   err))) {
         printf("Create handle error = %d\n", rc);
         return -1;
    }

    /* Connect with specified credentials       */
    if (0 != (rc = eimConnect(&handle,
                              con,
                              err))) {
         printf("Connect error = %d\n", rc);
         eimDestroyHandle(&handle, err);
         return -1;
    }

    /* Get all registries                       */
    memset(eimerr, 0x00, 200);
    err->memoryProvidedByCaller = 200;
    if (0 != (rc = eimListRegistries(&handle,
                                     NULL,
                                     NULL,
                                     EIM_ALL_REGISTRIES,
                                     16000,
                                     list,
```

```
                                    err))) {
        printf("List registries error = %d - %s\n", rc, eimErr2String(err));
        if (NULL != errstr) free(errstr);
        eimDestroyHandle(&handle, err);
        return -1;
    }


    /* Print the results                        */
    printListResults(list);

    /* Destroy the handle                       */
    rc = eimDestroyHandle(&handle, err);

    return 0;

}

void printListResults(EimList * list)
{
    int             i;
    EimRegistry    * entry;
    enum EimStatus * status;

    printf("___________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");


    entry = (EimRegistry *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("===============\n");
        printf("Entry %d.\n", i);

        /* Registry kind */
        printRegistryKind(entry->kind);

        /* Print out results */
        printListData("Registry Name",
                      entry,
                      offsetof(EimRegistry, name));
        printListData("Registry Type",
                      entry,
                      offsetof(EimRegistry, type));
        printListData("description",
                      entry,
                      offsetof(EimRegistry, description));
        printListData("entryUUID",
                      entry,
                      offsetof(EimRegistry, entryUUID));
        printListData("URI",
                      entry,
                      offsetof(EimRegistry, URI));
        printListData("system registry name",
                      entry,
                      offsetof(EimRegistry, systemRegistryName));
        printAliasSubList(entry,
                          offsetof(EimRegistry, registryAlias));

        status = (enum EimStatus *)((char *)entry +
                                    offsetof(EimRegistry, mappingLookup));
        if ( (*status) == EIM_STATUS_ENABLED) {
```

```
           printf("     Mapping Lookup operations are Enabled\n");
        } else {
           printf("     Mapping Lookup operations are Disabled\n");
        }

        status = (enum EimStatus *)((char *)entry +
                                    offsetof(EimRegistry, policyAssociations));
        if ( (*status) == EIM_STATUS_ENABLED) {
           printf("     Policy Associations are Enabled\n");
        } else {
           printf("     Policy Associations are Disabled\n");
        }

        /* advance to next entry */
        entry = (EimRegistry *)((char *)entry + entry->nextEntry);


    }
    printf("\n");


}
void printRegistryKind(int kind)
{
    switch(kind)
    {
        case EIM_SYSTEM_REGISTRY:
            printf("    System Registry.\n");
            break;
        case EIM_APPLICATION_REGISTRY:
            printf("Application Registry.\n");
            break;
        default:
            printf("ERROR - unknown registry kind.\n");
            break;
    }
}


void printListData(char * fieldName,
                   void * entry,
                   int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;


    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}



void printAliasSubList(void * entry,
                       int offset)
```

```
{
    int i;
    EimSubList * subList;
    EimRegistryAlias * subentry;


    /* Address the EimSubList object */
    subList = (EimSubList *)((char *)entry + offset);


    if (subList->listNum > 0)
    {


        subentry = (EimRegistryAlias *)((char *)entry +
                                        subList->disp);
        for (i = 0; i < subList->listNum; i++)
        {


            /* Print out results */
            printListData("Registry alias type",
                          subentry,
                          offsetof(EimRegistryAlias, type));
            printListData("Registry alias value",
                          subentry,
                          offsetof(EimRegistryAlias, value));


            /* advance to next entry */
            subentry = (EimRegistryAlias *)((char *)subentry +
                                            subentry->nextEntry);
        }
    }

}
```

eimListRegistryAliases

Purpose

Returns a list of all the aliases defined for a particular registry.

Format

```
#include <eim.h>


int eimListRegistryAliases(EimHandle      * eim,
                           char           * registryName,
                           unsigned int     lengthOfListData,
                           EimList        * listData,
                           EimRC          * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

registryName
(Input) The name of the registry for which to list aliases. Registry names are case-independent (meaning, not case-sensitive).

The following special characters are not allowed in registry names:

```
, = + < > # ; \ *
```

lengthOfListData
(Input) The number of bytes the caller provides for the *listData* parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The minimum size required is 20 bytes.

listData
(Output) A pointer to the data to return.

The EimList structure contains information about the returned data. The data returned is a linked list of EimRegistryAlias structures. The firstEntry field in the EimList structure is used to get to the first EimRegistryAlias structure in the linked list. The number of completed EimRegistryAlias structures is returned in entriesReturned. The bytesReturned variable has the number of bytes the API used for the returned entries. If the number of entries returned is less than the number of entries available, the returned data contains as many complete EimRegistryAlias structures as will fit. It can also contain a partial EimRegistryAlias structure.The EimList structure follows:

```
typedef struct EimList
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API                       */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API                          */
    unsigned int entriesReturned;   /* Number of entries actually
                                       returned by the API              */
    unsigned int entriesAvailable;  /* Number of entries available to be
                                       returned by the API              */
    unsigned int firstEntry;        /* Displacement to the first linked
```

```
                                                 list entry. This byte offset is
                                                 relative to the start of the
                                                 EimList structure.              */
        } EimList;
```

The EimRegistryAlias structure follows:

```
        typedef struct EimRegistryAlias
        {
            unsigned int nextEntry;         /* Displacement to next entry.  This
                                               byte offset is relative to the
                                               start of this structure         */
            EimListData type;               /* Alias type                      */
            EimListData value;              /* Alias value                     */
        } EimRegistryAlias;
```

The EimListData follows:

```
        typedef struct EimListData
        {
            unsigned int length;            /* Length of data                  */
            unsigned int disp;              /* Displacement to data.  This byte
                                               offset is relative to the start of
                                               the parent structure, i.e. the
                                               structure containing this
                                               structure.                      */
        } EimListData;
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimChangeRegistryAlias” on page 207
- “eimGetRegistryNameFromAlias” on page 265

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM identifiers administrator
- EIM registry *X* administrator
- EIM mapping lookup

The returned list contains only the information that the user has authority to access, meaning it could be empty.

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
EBADNAME	Registry not found or insufficient access to EIM data.	
	EIMERR_NOREG (28)	EIM registry not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_EIMLIST_SIZE (16)	Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example lists all aliases for the specified registry:

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>


void printListResults(EimList * list);
void printListData(char * fieldName, void * entry, int offset);
```

```
int main(int argc, char *argv[])
{
    int            rc;
    char           eimerr[200];
    EimRC        * err;
    EimHandle      handle;
    EimConnectInfo con;
    char         * ldapHost =
      "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";
    char           listData[1000];
    EimList      * list = (EimList * ) listData;

    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=administrator";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Create handle with specified LDAP URL    */
    if (0 != (rc = eimCreateHandle(&handle,
                                   ldapHost,
                                   err))) {
         printf("Create handle error = %d\n", rc);
         return -1;
    }

    /* Connect with specified credentials       */
    if (0 != (rc = eimConnect(&handle,
                              con,
                              err))) {
         printf("Connect error = %d\n", rc);
         eimDestroyHandle(&handle, err);
         return -1;
    }

    /* Get all aliases for the registry         */
    if (0 != (rc = eimListRegistryAliases(&handle,
                                          "MyRegistry",
                                          1000,
                                          list,
                                          err)))
    {
        printf("List registry aliases error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    rc = eimDestroyHandle(&handle, err);

    return 0;
}

void printListResults(EimList * list)
{
    int i;
    EimRegistryAlias * entry;
```

```
    printf("__________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");


    entry = (EimRegistryAlias *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {

        /* Print out results */
        printListData("Registry Alias Type",
                      entry,
                      offsetof(EimRegistryAlias, type));
        printListData("Registry Alias Value",
                      entry,
                      offsetof(EimRegistryAlias, value));


        /* advance to next entry */
        entry = (EimRegistryAlias *)((char *)entry + entry->nextEntry);


    }
    printf("\n");


}
void printListData(char * fieldName,
                   void * entry,
                   int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;


    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}
```

eimListRegistryAssociations

Purpose

Lists association information for the registry or domain.

Format

```
#include <eim.h>
int eimListRegistryAssociations(EimHandle              * eim,
                                enum EimAssociationType  associationType,
                                char                   * registryName,
                                char                   * registryUserName,
                                unsigned int             lengthOfListData,
                                EimList                * listData,
                                EimRC                  * eimrc)
```

Parameters

eim
(Input) The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

assocationType
(Input) The type of associations to be listed. Valid values are:

EIM_ALL_ASSOC (0)
List all associations.

EIM_TARGET (1)
List target associations. If this type is chosen, the following fields are returned:

- registryName
- registryUserName
- identifier
- targetMappingLookupStatus

Source associations found return the fields described under EIM_SOURCE and target associations found will return the fields listed under EIM_TARGET.

EIM_SOURCE (2)
List source associations. If this type is chosen, the following fields are returned:

- registryName
- registryUserName
- identifier
- sourceMappingLookupStatus

EIM_SOURCE_AND_TARGET (3)
List source and target associations. Source associations found will return the fields described under EIM_SOURCE, and target associations found will return the fields listed under EIM_TARGET

EIM_ADMIN (4)
List administrative associations. If this type is chosen, the following fields are returned:

- registryName
- registryUserName

- identifier

EIM_ALL_POLICY_ASSOC (5)
List all policy associations. This returns all of the following policy associations: EIM_CERT_FILTER_POLICY, EIM_DEFAULT_REG_POLICY, EIM_DEFAULT_DOMAIN_POLICY, and EIM_DEFAULT_DOMAIN_POLICY.

EIM_CERT_FILTER_POLICY (6)
List certificate filter policy associations. If this type is chosen, the following fields are returned:

- registryName
- registryUserName
- sourceRegistry
- filterValue
- domainPolicyAssocStatus
- sourceMappingLookupStatus
- targetMappingLookupStatus
- targetPolicyAssocStatus

EIM_DEFAULT_REG_POLICY (7)
List default registry policy associations. If this type is chosen, the following fields are returned:

- registryName
- registryUserName
- sourceRegistry
- domainPolicyAssocStatus
- sourceMappingLookupStatus
- targetMappingLookupStatus
- targetPolicyAssocStatus

EIM_DEFAULT_DOMAIN_POLICY (8)
List default domain policy associations. If this type is chosen, the following fields are returned:

- registryName
- registryUserName
- domainPolicyAssocStatus
- targetMappingLookupStatus
- targetPolicyAssocStatus

registryName
(Input) The name of the registry for which to list association information.

registryUserName
(Input) The name of the registry user name for which to list association information.

lengthOfListData
(Input) The number of bytes provided by the caller for the listData parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The minimum size is 20 bytes.

listData

(Output) A pointer to the EimList structure, which contains information about the returned data. The API will return as much data as space has allowed. The data returned is a linked list of EimRegistryAssociation structures. The EimList structure follows:

```
typedef struct EimList
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API                       */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API                          */
    unsigned int entriesReturned;   /* Number of entries actually
                                       returned by the API              */
    unsigned int entriesAvailable;  /* Number of entries available to be
                                       returned by the API              */
    unsigned int firstEntry;        /* Displacement to the first linked
                                       list entry. This byte offset is
                                       relative to the start of the
                                       EimList structure.               */
} EimList;
```

The EimRegistryAssociation structure follows:

```
typedef struct EimRegistryAssociation
  {
      unsigned int nextEntry;         /* Displacement to next entry.  This
                                         byte offset is relative to the
                                         start of this structure          */
      enum EimAssociationType type;   /* Type of association.             */
      EimListData registryName;       /* Registry name the association
                                         is defined to.                   */
      EimListData registryUserName;   /* Registry user name the
                                         association is defined to.       */
      EimListData identifier;         /* Unique name for eim identifier.*/
      EimListData sourceRegistry;     /* Source registry name the
                                         association is defined for.      */
      EimListData filterValue;        /* Filter value.                */
      enum EimPolicyStatus domainPolicyAssocStatus;
                                /* Policy association status for the domain:
                                        0 = not enabled
                                        1 = enabled                       */
      enum EimPolicyStatus sourceMappingLookupStatus;
                                /* Mapping lookup status for the
                                   source registry:
                                        0 = not enabled
                                        1 = enabled                       */
         enum EimPolicyStatus targetMappingLookupStatus;
                                /* Mapping lookup status for the
                                   target registry:
                                        0 = not enabled
                                        1 = enabled                       */
        enum EimPolicyStatus targetPolicyAssocStatus;
                                /* Policy association status for
                                   the target registry:
                                        0 = not enabled
                                        1 = enabled                       */
    } EimRegistryAssociation;
```

The EimListData structure follows:

```
typedef struct EimListData
  {
      unsigned int length;            /* Length of data                */
      unsigned int disp;              /* Displacement to data.  This byte
                                         offset is relative to the start of
```

```
                                                the parent structure; that is, the
                                                structure containing this
                                                structure.                      */
    } EimListData;
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddPolicyAssociation” on page 183
- “eimAddAssociation” on page 174
- “eimGetAssociatedIdentifiers” on page 254

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM identifiers administrator
- EIM mapping lookup
- EIM authority to an individual registry

The returned list contains only the information that the user has authority to access, meaning it could be empty.

z/OS authorization
No special authorization is needed.

Return Values

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	(z/OS does not return this value.) Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBADNAME	Registry name is not valid or insufficient access to EIM data..	
	EIMERR_NOREG (28)	EIM registry not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.

Return Value	Meaning
ECONVERT	Data conversion error.
	EIMERR_DATA_CONVERSION (13) (z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.
	EIMERR_ASSOC_TYPE_INVAL (4) Association type is not valid.
	EIMERR_EIMLIST_SIZE (16) Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_FUNCTION_NOT_SUPPORTED (70) The specified or configured EIM Domain controller does not support this API.
	EIMERR_HANDLE_INVAL (17) EimHandle is not valid.
	EIMERR_PARM_REQ (34) Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35) (z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41) Unexpected error accessing parameter.
ENOMEM	Unable to allocate required space.
	EIMERR_NOMEM (27) No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.
	EIMERR_NOT_CONN (31) Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EUNKNOWN	Unexpected exception.
	EIMERR_LDAP_ERR (23) Unexpected LDAP error.
	EIMERR_UNKNOWN (44) Unknown error or unknown system state.
	EIMERR_UNEXP_OBJ_VIOLATION (56) Unexpected object violation.

Example

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void printAssociationType(int type);
void printStatus(char * fieldname, enum EimStatus status);
void printListResults(EimList * list);
void printListData(char * fieldName, void * entry, int offset);

int main (int argc, char *argv[])
{
    int            rc;
    char           eimerr[200];
    EimRC        * err;
    EimHandle      handle;
    char           listData[4000];
    EimList      * list = (EimList *)listData;
```

```
    EimConnectInfo con;
    char         * ldapHost =
      "ldap://localhost:389/ibm-eimDomainName=MyDomain,o=MyCompany,c=us";

    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=administrator";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Create handle with specified LDAP URL    */
    if (0 != (rc = eimCreateHandle(&handle, ldapHost, err))) {
         printf("Create handle error = %d\n", rc);
         return -1;
    }

    /* Connect with specified credentials       */
    if (0 != (rc = eimConnect(&handle, con, err))) {
         printf("Connect error = %d\n", rc);
         eimDestroyHandle(&handle, err);
         return -1;
    }

    /* Get all Policy Associations              */
    if (0 != (rc = eimListRegistryAssociations(&handle,
                                               EIM_ALL_POLICY_ASSOC,
                                               NULL,
                                               NULL,
                                               4000,
                                               list,
                                               err)))
    {
        printf("List EIM Registry Associations error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    return 0;

}

void printListResults(EimList * list)
{
    int                      i;
    EimRegistryAssociation * entry;

    printf("___________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimRegistryAssociation *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("===============\n");
        printf("Entry %d.\n", i);
```

```
            /* Print out results */
            printAssociationType(entry->type);
            printListData("Registry Name",
                          entry,
                          offsetof(EimRegistryAssociation, registryName));
            printListData("Registry User Name",
                          entry,
                          offsetof(EimRegistryAssociation, registryUserName));
            printListData("EIM Identifier",
                          entry,
                          offsetof(EimRegistryAssociation, identifier));
            if (entry->type != EIM_DEFAULT_DOMAIN_POLICY) {
               printListData("Source Registry",
                             entry,
                             offsetof(EimRegistryAssociation, sourceRegistry));
            }
            if (entry->type == EIM_CERT_FILTER_POLICY) {
               printListData("Filter Value",
                             entry,
                             offsetof(EimRegistryAssociation, filterValue));
            }
            printStatus("Domain policy association status",
                        entry->domainPolicyAssocStatus);
            if (entry->type != EIM_DEFAULT_DOMAIN_POLICY) {
               printStatus("Source registry mapping lookup status",
                           entry->sourceMappingLookupStatus);
            }
            printStatus("Target registry mapping lookup status",
                        entry->targetMappingLookupStatus);
            printStatus("Target registry policy association status",
                        entry->targetPolicyAssocStatus);
            /* advance to next entry */
            /*entry += entry->nextEntry;*/
            entry = (EimRegistryAssociation *)((char *)entry + entry->nextEntry);
        }
        printf("\n");
    }

    void printAssociationType(int type)
    {
        switch(type)
        {
            case EIM_TARGET:
                printf("    Target Association.\n");
                break;
            case EIM_SOURCE:
                printf("    Source Association.\n");
                break;
            case EIM_ADMIN:
                printf("    Administrative Association.\n");
                break;
            case EIM_CERT_FILTER_POLICY:
                printf("    Certificate Filter Policy Association.\n");
                break;
            case EIM_DEFAULT_REG_POLICY:
                printf("    Default Registry Policy Association.\n");
                break;
            case EIM_DEFAULT_DOMAIN_POLICY:
                printf("    Default Domain Policy Association.\n");
                break;
            default:
                printf("ERROR - unknown policy association type(%d).\n", type);
                break;
        }
    }

    void printStatus(char * fieldName,
```

```
                enum EimStatus status)
{

    printf("     %s = ",fieldName);

    switch(status)
    {
        case EIM_STATUS_NOT_ENABLED:
            printf("    Not enabled.\n");
            break;
        case EIM_STATUS_ENABLED:
            printf("    Enabled.\n");
            break;
        default:
            printf("ERROR - unknown status value.\n");
            break;
    }
}

void printListData(char * fieldName,
                   void * entry,
                   int offset)
{
    EimListData * listData;
    char        * data;
    int           dataLength;

    printf("     %s = ",fieldName);

    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
       printf("%.*s\n",dataLength, data);
    else
       printf("Not found.\n");

}
```

eimListRegistryUsers

Purpose

Lists the users in a particular registry that have target associations defined.

Format

```
#include <eim.h>


int eimListRegistryUsers(EimHandle      * eim,
                         char           * registryName,
                         char           * registryUserName,
                         unsigned int     lengthOfListData,
                         EimList        * listData,
                         EimRC          * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

registryName
(Input) The name of the registry that contains this user. Registry names are case-independent (meaning, not case-sensitive).

The following special characters are not allowed in registry names:

```
, = + < > # ; \ *
```

registryUserName
(Input) The name of the user to list in this registry. NULL indicates listing all users. The registry user name should begin with a non-blank character.

lengthOfListData
(Input) The number of bytes the caller provides for the listData parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The minimum size required is 20 bytes.

listData
(Output) A pointer to the EimList structure. The EimList structure contains information about the returned data. The data returned is a linked list of EimRegistryUser structures. The firstEntry field in the EimList structure is used to get to the first EimRegistryUser structure in the linked list. The number of completed EimRegistryUser structures is returned in entriesReturned. The bytesReturned variable has the number of bytes the API used for the returned entries. If the number of entries returned is less than the number of entries available, the returned data contains as many complete EimRegistryUser structures as will fit. It can also contain a partial EimRegistryUser structure.The EimList structure follows:

```
typedef struct EimList
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API                       */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API                          */
    unsigned int entriesReturned;   /* Number of entries actually
```

```
                                    returned by the API             */
    unsigned int entriesAvailable;  /* Number of entries available to be
                                    returned by the API             */
    unsigned int firstEntry;        /* Displacement to the first linked
                                    list entry. This byte offset is
                                    relative to the start of the
                                    EimList structure.              */
} EimList;
```

The EimRegistryUser structure follows:

```
typedef struct EimRegistryUser
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                    byte offset is relative to the
                                    start of this structure.        */
    EimListData registryUserName;   /* Name                            */
    EimListData description;        /* Description                     */
    EimSubList  additionalInfo;     /* EimAddlInfo sublist             */
} EimRegistryUser;
```

Registry users might have defined several additional attributes. In the EimRegistryUser structure, additionalInfo gives addressability to the first EimAddlInfo structure that contains a linked list of attributes. The EimAddlInfo structure follows:

```
typedef struct EimAddlInfo
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                    byte offset is relative to the
                                    start of this structure.        */
    EimListData addlInfo;           /* Additional info                 */
} EimAddlInfo;
```

The EimSubList structure follows:

```
typedef struct EimSubList
{
    unsigned int listNum;           /* Number of entries in the list  */
    unsigned int disp;              /* Displacement to sublist. This
                                    byte offset is relative to the
                                    start of the parent structure, i.e.
                                    the structure containing this
                                    structure.                      */
} EimSubList;
```

The EimListData structure follows:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                  */
    unsigned int disp;              /* Displacement to data.  This byte
                                    offset is relative to the start of
                                    the parent structure, i.e. the
                                    structure containing this
                                    structure.                      */
} EimListData;
```

eimrc

(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- "eimChangeRegistryUser" on page 211

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM identifiers administrator
- EIM registry *X* administrator
- EIM mapping lookup

The list returned contains only the information that the user has authority to access.

z/OS authorization

No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful. Check the entriesReturned member of the listData to determine if any entries were returned.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
EBADNAME	Registry not found or insufficient access to EIM data.	
	EIMERR_NOREG (28)	EIM registry not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.

Return Value	Meaning	
EINVAL	Input parameter was not valid.	
	EIMERR_EIMLIST_SIZE (16)	Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example lists all users in the specified registry:

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>


void printListResults(EimList * list);
void printSubListData(char * fieldName, void * entry, int offset);
void printListData(char * fieldName, void * entry, int offset);

int main(int argc, char *argv[])
{
    int            rc;
    char           eimerr[200];
    EimRC        * err;
    EimHandle      handle;
    EimConnectInfo con;
    char         * ldapHost =
       "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";
    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=admin";
```

```
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Create handle with specified LDAP URL    */
    if (0 != (rc = eimCreateHandle(&handle,
                                   ldapHost,
                                   err))) {
         printf("Create handle error = %d\n", rc);
         return -1;
    }

    /* Connect with specified credentials       */
    if (0 != (rc = eimConnect(&handle,
                              con,
                              err))) {
         printf("Connect error = %d\n", rc);
         eimDestroyHandle(&handle, err);
         return -1;
    }

    /* Get registry user                        */
    if (0 != (rc = eimListRegistryUsers(&handle,
                                        "MyRegistry",
                                        NULL,
                                        1000,
                                        list,
                                        err)))
    {
        printf("List registry users error = %d\n", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    /* Destroy the handle                       */
    rc = eimDestroyHandle(&handle, err);

    return 0;
}

void printListResults(EimList * list)
{
    int i;
    EimRegistryUser * entry;


    printf("__________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimRegistryUser *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("===============\n");
        printf("Entry %d.\n", i);

        /* Print out results */
        printListData("Registry user name",
                      entry,
                      offsetof(EimRegistryUser, registryUserName));
        printListData("description",
                      entry,
```

```
                    offsetof(EimRegistryUser, description));
        printSubListData("Additional information",
                    entry,
                    offsetof(EimRegistryUser, additionalInfo));

        /* advance to next entry */
        entry = (EimRegistryUser *)((char *)entry + entry->nextEntry);

    }
    printf("\n");

}
void printSubListData(char * fieldName, void * entry, int offset)
{
    int i;
    EimSubList * subList;
    EimAddlInfo * subentry;

    /* Address the EimSubList object */
    subList = (EimSubList *)((char *)entry + offset);

    if (subList->listNum > 0)
    {
        subentry = (EimAddlInfo *)((char *)entry + subList->disp);
        for (i = 0; i < subList->listNum; i++)
        {

            /* Print out results */
            printListData(fieldName,
                          subentry,
                          offsetof(EimAddlInfo, addlInfo));

            /* advance to next entry */
            subentry = (EimAddlInfo *)((char *)subentry +
                                       subentry->nextEntry);
        }
    }

}

void printListData(char * fieldName, void * entry, int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}
```

eimListUserAccess

Purpose

Lists the access groups of which the given user is a member.

Format

```
#include <eim.h>


int eimListUserAccess(EimHandle      * eim,
                      EimAccessUser  * accessUser,
                      unsigned int     lengthOfListData,
                      EimList        * listData,
                      EimRC          * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

accessUser
(Input) A structure that contains the user information for which to retrieve access.

EIM_ACCESS_DN Indicates a distinguished name defined in an LDAP directory that can be used to bind to the EIM domain.

EIM_ACCESS_LOCAL_USER
(z/OS does not support this; for RACF user IDs, use EIM_ACCESS_DN instead.) EIM_ACCESS_LOCAL_USER indicates a local user name on the system where the API runs. EIM converts the local user name to the appropriate access ID for this system.

EIM_ACCESS_KERBEROS Indicates a Kerberos principal. EIM converts the Kerberos principal to the appropriate access ID, for example, converting `petejones@therealm` to `ibm-kn=petejones@threalm`.

The EimAccessUser structure layout follows:

```
enum EimAccessUserType {
    EIM_ACCESS_DN,
    EIM_ACCESS_KERBEROS,
    EIM_ACCESS_LOCAL_USER
};


typedef struct EimAccessUser
{
    union {
        char * dn;
        char * kerberosPrincipal;
        char * localUser;
    } user;
    enum EimAccessUserType userType;
} EimAccessUser;
```

lengthOfListData
(Input) The number of bytes the caller provides for the listData parameter. If the value of bytesReturned is less than bytesAvailable in the returned listData structure, you can use this number as the bytesAvailable size, update the lengthOfListData parameter, and reissue the API to retrieve the data. The minimum size required is 20 bytes.

listData
(Output) A pointer to the EimList structure. The EimList structure contains information about the returned data. The data returned is a linked list of EimUserAccess structures. The firstEntry field in the EimList structure is used to get to the first EimUserAccess structure in the linked list. The number of completed EimUserAccess structures is returned in entriesReturned. The bytesReturned variable has the number of bytes the API used for the returned entries. If the number of entries returned is less than the number of entries available, the returned data contains as many complete EimUserAccess structures as will fit. It can also contain a partial EimUserAccess structure, but not a partial entry.The EimList structure follows:

```
typedef struct EimList
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API.                      */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API.                         */
    unsigned int entriesReturned;   /* Number of entries actually
                                       returned by the API.             */
    unsigned int entriesAvailable;  /* Number of entries available to be
                                       returned by the API.             */
    unsigned int firstEntry;        /* Displacement to the first linked
                                       list entry. This byte offset is
                                       relative to the start of the
                                       EimList structure.               */
} EimList;
```

The EimUserAccess structure follows:

```
typedef struct EimUserAccess
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                start of this structure.         */
    enum EimAccessIndicator eimAdmin;
    enum EimAccessIndicator eimRegAdmin;
    enum EimAccessIndicator eimIdenAdmin;
    enum EimAccessIndicator eimMappingLookup;
    EimSubList  registries;         /* EimRegistryName sublist        */
} EimUserAccess;
```

The registries EimSubList gives addressability to a linked list of EimRegistryName structures. The EimRegistryName structure follows:

```
typedef struct EimRegistryName
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure.         */
    EimListData name;               /* Name                           */
} EimRegistryName;
```

The EimSubList structure follows:

```
typedef struct EimSubList
{
    unsigned int listNum;          /* Number of entries in the list  */
    unsigned int disp;             /* Displacement to sublist. This
                                      byte offset is relative to the
                                      start of the parent structure, i.e.
                                      the structure containing this
                                      structure.                     */
} EimSubList;
```

The EimListData structure follows:

```
typedef struct EimListData
{
    unsigned int length;           /* Length of data                 */
    unsigned int disp;             /* Displacement to data.  This byte
                                      offset is relative to the start of
                                      the parent structure, i.e. the
                                      structure containing this
                                      structure.                     */
} EimListData;
```

eimrc
(Input) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddAccess” on page 166
- “eimListAccess” on page 286
- “eimRemoveAccess” on page 355
- “eimQueryAccess” on page 351

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

The list returned contains only the information that the user has authority to access.

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning
0	Request was successful.
EACCES	Access denied. Not enough permissions to access data.
EBADDATA	eimrc is not valid.

Return Value	Meaning	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_ACCESS_USERTYPE_INVAL (3)	Access user type is not valid.
	EIMERR_EIMLIST_SIZE (16)	Length of EimList is not valid. EimList must be at least 20 bytes in length.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example will list the access for the user with distinguished name ″cn=pete,o=ibm,c=us″.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>


void printListResults(EimList * list);
void printSubListData(char * fieldName, void * entry, int offset);
void printListData(char * fieldName, void * entry, int offset);

int main(int argc, char *argv[])
{
    int            rc;
    char           eimerr[200];
    EimRC        * err;
    EimHandle      handle;
    EimAccessUser  user;
    EimConnectInfo con;
    char           listData[1000];
```

```
    EimList      * list = (EimList * ) listData;
    char         * ldapHost =
      "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";

    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;

    con.type = EIM_SIMPLE;
    con.creds.simpleCreds.protect = EIM_PROTECT_NO;
    con.creds.simpleCreds.bindDn = "cn=admin";
    con.creds.simpleCreds.bindPw = "secret";
    con.ssl = NULL;

    /* Create handle with specified LDAP URL    */
    if (0 != (rc = eimCreateHandle(&handle,
                                   ldapHost,
                                   err))) {
        printf("Create handle error = %d\n", rc);
        return -1;
    }

    /* Connect with specified credentials       */
    if (0 != (rc = eimConnect(&handle,
                              con,
                              err))) {
        printf("Connect error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    /* Set up access user information           */
    user.userType = EIM_ACCESS_DN;
    user.user.dn  = "cn=pete,o=ibm,c=us";

    /* Get user accesses                        */
    if (0 != (rc = eimListUserAccess(&handle,
                                     &user,
                                     1000,
                                     list,
                                     err)))
    {
        printf("List user access error = %d\n", rc);
        eimDestroyHandle(&handle, err);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    /* Destroy the handle                       */
    rc = eimDestroyHandle(&handle, err);

    return 0;
}

void printListResults(EimList * list)
{
    int i;
    EimUserAccess * entry;
    EimListData * listData;
    EimRegistryName * registry;

    printf("___________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
```

```
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    if (list->entriesReturned > 1)
        printf("Unexpected number of entries returned.\n");

    entry = (EimUserAccess *)((char *)list + list->firstEntry);
    if (EIM_ACCESS_YES == entry->eimAdmin)
        printf("     EIM Admin.\n");
    if (EIM_ACCESS_YES == entry->eimRegAdmin)
        printf("     EIM Reg Admin.\n");
    if (EIM_ACCESS_YES == entry->eimIdenAdmin)
        printf("     EIM Iden Admin.\n");
    if (EIM_ACCESS_YES == entry->eimMappingLookup)
        printf("     EIM Mapping Lookup.\n");

    printf("     Registries:\n");
    printSubListData("Registry names",
                      entry,
                      offsetof(EimUserAccess, registries));
    printf("\n");

}

void printSubListData(char * fieldName, void * entry, int offset)
{
    int i;
    EimSubList * subList;
    EimRegistryName * subentry;

    /* Address the EimSubList object */
    subList = (EimSubList *)((char *)entry + offset);

    if (subList->listNum > 0)
    {
        subentry = (EimRegistryName *)((char *)entry + subList->disp);
        for (i = 0; i < subList->listNum; i++)
        {
            /* Print out results */
            printListData(fieldName,
                          subentry,
                          offsetof(EimRegistryName, name));

            /* advance to next entry */
            subentry = (EimRegistryName *)((char *)subentry +
                                       subentry->nextEntry);
        }
    }
}

void printListData(char * fieldName, void * entry, int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
```

```
            printf("%.*s\n",dataLength, data);
        else
            printf("Not found.\n");
    }
```

eimQueryAccess

Purpose

Queries to check if the user has the specified access.

Format

```
#include <eim.h>

int eimQueryAccess(EimHandle          * eim,
                   EimAccessUser      * accessUser,
                   enum EimAccessType   accessType,
                   char               * registryName,
                   unsigned int       * accessIndicator,
                   EimRC              * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

accessUser
(Input) A structure that contains the user information for which to query access.

EIM_ACCESS_DN Indicates a distinguished name defined in an LDAP directory that you can use to bind to the EIM domain.

EIM_ACCESS_LOCAL_USER
(z/OS does not support this. Use EIM_ACCESS_DN instead.) It indicates a local user name on the system where the API runs. The local user name is converted to the appropriate access ID for this system.

EIM_ACCESS_KERBEROS Indicates a Kerberos identity. The Kerberos identity is converted to the appropriate access ID. For example, EIM converts `petejones@therealm` to `ibm-kn=petejones@threalm`.

The EimAccessUser structure layout follows:

```
enum EimAccessUserType {
    EIM_ACCESS_DN,
    EIM_ACCESS_KERBEROS,
    EIM_ACCESS_LOCAL_USER
};


typedef struct EimAccessUser
{
    union {
        char * DN;
        char * kerberosPrincipal;
        char * localUser;
    } user;
    enum EimAccessUserType userType;
} EimAccessUser;
```

accessType
(Input) The type of access to check. Valid values are:

EIM_ACCESS_ADMIN (0) Administrative authority to the entire EIM domain.

EIM_ACCESS_REG_ADMIN (1)
Administrative authority to all registries in the EIM domain.

EIM_ACCESS_REGISTRY (2)
Administrative authority to the registry specified in the registryName parameter.

EIM_ACCESS_IDENTIFIER_ADMIN (3)
Administrative authority to all of the identifiers in the EIM domain.

EIM_ACCESS_MAPPING_LOOKUP (4)
Authority to perform mapping lookup operations.

registryName
(Input) The name of the EIM registry for which to check the access. Registry names are case-independent (not case-sensitive). This parameter is used only if accessType is EIM_ACCESS_REGISTRY. If accessType is anything other than EIM_ACCESS_REGISTRY, this parameter must be NULL.

The following special characters are not allowed in registry names:

```
, = + < > # ; \ *
```

accessIndicator
(Output) Indicates whether access is found.

EIM_ACCESS_NO (0) Access not found.

EIM_ACCESS_YES (1) Access found.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddAccess” on page 166
- “eimListAccess” on page 286
- “eimListUserAccess” on page 344
- “eimRemoveAccess” on page 355

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

z/OS authorization

No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_ACCESS_TYPE_INVAL (2)	Access type is not valid.
	EIMERR_ACCESS_USERTYPE_INVAL (3)	Access user type is not valid.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_REG_MUST_BE_NULL (55)	Registry name must be NULL when access type is not EIM_ACCESS_REGISTRY.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following illustrates a query to see if the distinguished name ″cn=pete,o=ibm,c=us″ is a member of the ″EIM Administrator″ access group.

```
#include <eim.h>


.
 .
 .
   int           rc;
   char          eimerr[200];
   EimRC       * err;
   EimHandle     handle;
   EimAccessUser user;
   unsigned int indicator;
 .
 .
 .
   /* Set up error structure.                  */
   memset(eimerr,0x00,200);
   err = (EimRC *)eimerr;
   err->memoryProvidedByCaller = 200;
 .
 .
 .
   /* Set up access user info                  */
   user.userType = EIM_ACCESS_DN;
   user.user.DN="cn=pete,o=ibm,c=us";

   /* Query access for this user.              */
   rc = eimQueryAccess(&handle,
                       &user,
                       EIM_ACCESS_ADMIN,
                       NULL,
                       &indicator,
                       err);
 .
 .
 .
```

eimRemoveAccess

Purpose

Removes the user from an EIM access group.

Format

```
#include <eim.h>


int eimRemoveAccess(EimHandle          * eim,
                    EimAccessUser      * accessUser,
                    enum EimAccessType   accessType,
                    char               * registryName,
                    EimRC              * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

accessUser
(Input) A structure that contains the user information from which to remove access.

EIM_ACCESS_DN Indicates a distinguished name defined in an LDAP directory that you can use to bind to the EIM domain.

EIM_ACCESS_LOCAL_USER
(z/OS does not support this. Use EIM_ACCESS_DN instead.) It indicates a local user name on the system where the API runs. The local user name is converted to the appropriate access ID for this system.

EIM_ACCESS_KERBEROS Indicates a Kerberos identity. The Kerberos identity is converted to the appropriate access ID. For example, EIM converts `petejones@therealm` to `ibm-kn=petejones@threalm`.

The EimAccessUser structure layout follows:

```
enum EimAccessUserType {
    EIM_ACCESS_DN,
    EIM_ACCESS_KERBEROS,
    EIM_ACCESS_LOCAL_USER
};


typedef struct EimAccessUser
{
    union {
        char * DN;
        char * kerberosPrincipal;
        char * localUser;
    } user;
    enum EimAccessUserType userType;
} EimAccessUser;
```

accessType
(Input) The type of access to remove. Valid values are:

EIM_ACCESS_ADMIN (0) Administrative authority to the entire EIM domain.

EIM_ACCESS_REG_ADMIN (1)
Administrative authority to all registries in the EIM domain.

EIM_ACCESS_REGISTRY (2)
Administrative authority to the registry specified in the registryName parameter.

EIM_ACCESS_IDENTIFIER_ADMIN (3)
Administrative authority to all of the identifiers in the EIM domain.

EIM_ACCESS_MAPPING_LOOKUP (4)
Authority to perform mapping lookup operations.

registryName
(Input) The name of the registry from which to remove access. Registry names are case-independent (meaning, not case-sensitive). This parameter is used only if accessType is EIM_ACCESS_REGISTRY. If accessType is anything other than EIM_ACCESS_REGISTRY, this parameter must be NULL.

The following special characters are not allowed in registry names:

```
, = + < > # ; \ *
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddAccess” on page 166
- “eimListAccess” on page 286
- “eimListUserAccess” on page 344
- “eimQueryAccess” on page 351

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_ACCESS_TYPE_INVAL (2)	Access type is not valid.
	EIMERR_ACCESS_USERTYPE_INVAL (3)	Access user type is not valid.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_REG_MUST_BE_NULL (55)	Registry name must be NULL when access type is not EIM_ACCESS_REGISTRY.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete the operation. Use the eimConnectToMaster API to get a write connection.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following illustrates removing the distinguished name(DN) of a user from the EIM Administror access group:

```
#include <eim.h>


.
 .
 .
    int           rc;
    char          eimerr[200];
    EimRC       * err;
    EimHandle     handle;
    EimAccessUser user;
 .
 .
 .
    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;
 .
 .
 .
    /* Set user information                     */
    user.userType = EIM_ACCESS_DN;
    user.user.DN="cn=pete,o=ibm,c=us";

    /* Remove access for this user.             */
    rc = eimRemoveAccess(&handle,
                         &user,
                         EIM_ACCESS_ADMIN,
                         NULL,
                         err);
 .
 .
 .
```

eimRemoveAssociation

Purpose

Removes an association for a user in a specified user registry with an EIM identifier.

Format

```
#include <eim.h>


int eimRemoveAssociation(EimHandle              * eim,
                         enum EimAssociationType  associationType,
                         EimIdentifierInfo      * idName,
                         char                   * registryName,
                         char                   * registryUserName,
                         EimRC                  * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

associationType
(Input) The type of association to remove. Valid values are:

EIM_ALL_ASSOC (0) Remove all associations.

EIM_TARGET (1) Remove a target association.

EIM_SOURCE (2) Remove a source association.

EIM_SOURCE_AND_TARGET (3)
Remove both a source association and a target association.

EIM_ADMIN (4) Remove an administrative association.

idName
(Input) A structure that contains the identifier name from which to remove this association. The layout of the EimIdentifierInfo structure follows:

```
enum EimIdType {
    EIM_UNIQUE_NAME,
    EIM_ENTRY_UUID,
    EIM_NAME
};


typedef struct EimIdentifierInfo
{
    union {
        char       * uniqueName;
        char       * entryUUID;
        char       * name;
    } id;
    enum EimIdType        idtype;
} EimIdentifierInfo;
```

`idtype`
The `idtype` in the EimIdentifierInfo structure indicates which identifier name has been provided. EIM_UNIQUE_NAME finds at most one matching

identifier. EIM_NAME results in an error if your EIM domain has more than one identifier containing the same name.

registryName
(Input) The registry name. Registry names are case-independent (meaning, not case-sensitive).

The following special characters are not allowed in registry names:

```
, = + < > # ; \ *
```

registryUserName
(Input) The registry user name. This can be normalized according to the normalization method for defined registry. The registry user name should begin with a non-blank character.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddAssociation” on page 174
- “eimGetAssociatedIdentifiers” on page 254
- “eimListAssociations” on page 291

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The authority that the access group has to the EIM data depends on the type of association being removed.

For administrative and source associations, the access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM identifiers administrator

For target associations, the access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM registries administrator
- EIM registry *X* administrator

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning
0	Request was successful.

Return Value	Meaning	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBADNAME	Registry or identifier name is not valid or insufficient access to EIM data.	
	EIMERR_IDNAME_AMBIGUOUS (20)	More than one EIM identifier was found that matches the requested identifier name.
	EIMERR_NOIDENTIFIER (25)	EIM identifier not found or insufficient access to EIM data.
	EIMERR_NOREG (28)	EIM registry not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_ASSOC_TYPE_INVAL (4)	Association type is not valid.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_IDNAME_TYPE_INVAL (52)	The EimIdType value is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
EMVSERR	An MVS environment or internal error has occurred.	
	EIMERR_ZOS_DATA_CONVERSION (6011)	Error occurred when converting data between code pages.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete the operation. Use the eimConnectToMaster API to get a write connection.

Return Value	Meaning	
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following illustrates remoing an administrative, source, and target association for a specified identifier:

```
#include <eim.h>
#include <stdio.h>


.
 .
 .
    int               rc;
    char              eimerr[200];
    EimRC           * err;
    EimHandle         handle;
    EimIdentifierInfo x;

    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;
 .
 .
 .
    /* Set up identifier information.           */
    x.idtype = EIM_UNIQUE_NAME;
    x.id.uniqueName = "mjones";

    /* Remove an Admin association        */
    rc = eimRemoveAssociation(&handle,
                              EIM_ADMIN,
                              &x,
                              "MyRegistry",
                              "maryjones",
                              err);
 .
 .
 .
    /* Remove a source association        */
    rc = eimRemoveAssociation(&handle,
                              EIM_SOURCE,
                              &x,
                              "kerberosRegistry",
                              "mjjones",
                              err);
 .
 .
 .
    /* Remove a target association        */
    rc = eimRemoveAssociation(&handle,
                              EIM_TARGET,
                              &x,
                              "MyRegistry",
                              "maryjo",
```

```
                                        err);
.
.
.
```

eimRemoveIdentifier

Purpose

Removes an EIM identifier and all of its associated mappings from the EIM domain.

Format

```
#include <eim.h>


int eimRemoveIdentifier(EimHandle         * eim,
                        EimIdentifierInfo * idName,
                        EimRC             * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

idName
(Input) A structure that contains the name for this identifier. EIM_NAME returns either one matching identifier or an error if your EIM domain has more than one identifier with the same non-unique name. The layout of the EimIdentifierInfo structure follows:

```
enum EimIdType {
    EIM_UNIQUE_NAME,
    EIM_ENTRY_UUID,
    EIM_NAME
};


typedef struct EimIdentifierInfo
{
    union {
        char       * uniqueName;
        char       * entryUUID;
        char       * name;
    } id;
    enum EimIdType        idtype;
} EimIdentifierInfo;
```

`idtype`
The `idtype` in the EimIdentifierInfo structure indicates which identifier name has been provided. EIM_UNIQUE_NAME finds at most one matching identifier. EIM_NAME results in an error if your EIM domain has more than one identifier containing the same name.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddIdentifier” on page 179
- “eimChangeIdentifier” on page 199
- “eimGetAssociatedIdentifiers” on page 254

- “eimListIdentifiers” on page 305

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

z/OS authorization

No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBADNAME	Identifier not found or insufficient access to EIM data.	
	EIMERR_IDNAME_AMBIGUOUS (20)	More than one EIM identifier was found that matches the requested identifier name.
	EIMERR_NOIDENTIFIER (25)	EIM identifier not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.)Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_IDNAME_TYPE_INVAL (52)	The EimIdType value is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.

Return Value	Meaning	
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete the operation. Use the eimConnectToMaster API to get a write connection.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates removing an EIM identifier:

```
#include <eim.h>


int           rc;
    char          eimerr[200];
    EimRC       * err;
    EimHandle     handle;
    EimIdentifierInfo idInfo;

    /* Set up error structure.                  */
    memset(eimerr,0x00,200);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 200;
 .
 .
 .

    /* Set identifier information.              */
    idInfo.idtype = EIM_UNIQUE_NAME;
    idInfo.id.uniqueName = "Mary Smith";

    /* Remove this identifier.                  */
    rc = eimRemoveIdentifier(&handle,
                             &idInfo,
                             err);
 .
 .
 .
```

eimRemovePolicyAssociation

Purpose

Removes the specified policy association from the domain.

Format

```
#include <eim.h>
int eimRemovePolicyAssociation(EimHandle                 * eim,
                               EimPolicyAssociationInfo  * policyAssoc,
                               EimRC                     * eimrc)
```

Parameters

eim

(Input) The EIM handle returned by a previous call to eimCreateHandle. A valid connection is required for this function.

policyAssoc

(Input) The information about the policy association to be removed. This field must contain information specific to the type of policy association you wish to remove. For example, for EIM_CERT_FILTER_POLICY (5) association type, the policyAssociation field must contain an EimCertificateFilterPolicyAssociation structure, for EIM_DEFAULT_REG_POLICY (6) association type, the policyAssociation field must contain an EimDefaultRegistryPolicyAssociation structure, and for EIM_DEFAULT_DOMAIN_POLICY (7) association type, the policyAssociation field must contain an EimDefaultDomainPolicyAssociation structure. The structure layouts follow:

```
enum EimAssociationType {
      EIM_ALL_ASSOC,                 /* Not supported on this interface*/
      EIM_TARGET,                    /* Not supported on this interface*/
      EIM_SOURCE,                    /* Not supported on this interface*/
      EIM_SOURCE_AND_TARGET,         /* Not supported on this interface*/
      EIM_ADMIN,                     /* Not supported on this interface*/
      EIM_ALL_POLICY_ASSOC,          /* Not supported on this interface*/
      EIM_CERT_FILTER_POLICY,        /* Association is a certificate
                                        filter policy association.     */
      EIM_DEFAULT_REG_POLICY,        /* Association is a default
                                        registry policy association    */
      EIM_DEFAULT_DOMAIN_POLICY      /* Policy is a default policy for
                                        the domain.                    */
  };
  typedef struct EimCertificateFilterPolicyAssociation
  {
      char  * sourceRegistry;        /* The source registry to remove
                                        the policy association from.   */
      char  * filterValue;           /* The filter value of the policy.*/
      char  * targetRegistry;        /* The name of the target registry
                                        that the filter value is mapped
                                        to.                            */
      char  * targetRegistryUserName; /* The name of the target registry
                                        user name that the filter value
                                        is mapped to.                  */
  } EimCertificateFilterPolicyAssociation;
  typedef struct EimDefaultRegistryPolicyAssociation
  {
      char  * sourceRegistry;        /* The source registry to remove
                                        the policy association from.   */
      char  * targetRegistry;        /* The name of the target registry
                                        that the policy association is
                                        mapped to.                     */
```

```
    char  * targetRegistryUserName; /* The name of the target registry
                                       user name that the policy
                                       association is mapped to.      */
} EimDefaultRegistryPolicyAssociation;
typedef struct EimDefaultDomainPolicyAssociation
{
    char  * targetRegistry;         /* The name of the target registry
                                       that the policy association is
                                       mapped to.                     */
    char  * targetRegistryUserName; /* The name of the target registry
                                       user name that the policy
                                       association is mapped to.      */
} EimDefaultDomainPolicyAssociation;
typedef struct EimPolicyAssociationInfo
{
    enum EimAssociationType type;
    union {
        EimCertificateFilterPolicyAssociation   certFilter;
        EimDefaultRegistryPolicyAssociation     defaultRegistry;
        EimDefaultDomainPolicyAssociation       defaultDomain;
    } policyAssociation;
} EimPolicyAssociationInfo;
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddPolicyAssociation” on page 183
- “eimListRegistryAssociations” on page 330

Authorization

EIM data
EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator
- EIM registries administrator
- EIM authority to an individual registry. This authority is needed to the target registry.

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning
0	Request was successful.
EACCES	Access denied. Not enough permissions to access data.
	EIMERR_ACCESS (1) Insufficient access to EIM data.

Return Value	Meaning	
EBADDATA	eimrc is not valid.	
EBADNAME	Registry not found or insufficient access to EIM data.	
	EIMERR_NOREG (28)	EIM registry not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_ASSOC_TYPE_INVAL (4)	Association type is not valid.
	EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified or configured EIM Domain controller does not support this API.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete this operation. Use eimConnectToMaster to get a write connection or use the URL for the master EIM domain controller which is writeable..
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.
	EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.

Example

```
#include <eim.h>
#include <string.h>
.
.
.
```

```
    int                       rc;
    char                      eimerr[250];
    EimRC                   * err;
    EimHandle                 handle;
    EimPolicyAssociationInfo assocInfo;

    /* Set up error structure.                  */
    memset(eimerr,0x00,250);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 250;
.
.
.
    /* Set up policy association information    */
    assocInfo.type = EIM_DEFAULT_REG_POLICY;
    assocInfo.policyAssociation.defaultRegistry.sourceRegistry = "MySourceRegistry";
    assocInfo.policyAssociation.defaultRegistry.targetRegistry = "localRegistry";
    assocInfo.policyAssociation.defaultRegistry.targetRegistryUserName = "mjjones";

    /* Remove the policy                        */
    rc = eimRemovePolicyAssociation(&handle, &assocInfo, err);
.
.
.
```

eimRemovePolicyFilter

Purpose

Removes the specified policy filter from the domain. When a policy filter is removed, all policy associations to the policy filter are also removed.

Format

```
#include <eim.h>
int eimRemovePolicyFilter(EimHandle            * eim,
                          EimPolicyFilterInfo  * filterInfo,
                          EimRC                * eimrc)
```

Parameters

eim

(Input) The EIM handle returned by a previous call to eimCreateHandle. A valid connection is required for this function.

filterInfo

The information about the policy filter to be removed. The structure layout follows:

```
enum EimPolicyFilterType {
       EIM_ALL_FILTERS,               /* All policy filters -- not
                                         supported for this interface.  */
       EIM_CERTIFICATE_FILTER         /* Policy filter is a certificate
                                         filter.                        */
   };
   typedef struct EimCertificatePolicyFilter
   {
       char  * sourceRegistry;        /* The source registry to remove the
                                         policy filters from.           */
       char  * filterValue;           /* The policy filter value. A NULL
                                         value will remove all policy
                                         filter values from the registry*/
   } EimCertificatePolicyFilter;
   typedef struct EimPolicyFilterInfo
   {
       enum EimPolicyFilterType type;
       union {
           EimCertificatePolicyFilter   certFilter;
       } filter;
   } EimPolicyFilterInfo;
```

eimrc

(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddPolicyFilter” on page 187
- “eimListPolicyFilters” on page 312

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also

have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

z/OS authorization

No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBADNAME	Registry not valid or insufficient access to EIM data.	
	EIMERR_NOREG (28)	EIM registry not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified or configured EIM Domain controller does not support this API.
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_POLICY_FILTER_TYPE_INVAL (60)	Policy filter type is not valid.
	EIMERR_REGTYPE_INVAL (62)	Registry type is not valid.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.

Return Value	Meaning	
EROFS	LDAP connection is for read-only. Need to connect to master. Use eimConnectToMaster API to get a write connection or use the URL for the master EIM domain controller which is writeable.	
	EIMERR_READ_ONLY (36)	This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete this operation. Use eimConnectToMaster to get a write connection.
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.
	EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.

Example

```
#include <eim.h>
#include <string.h>
.
.
.
    int                 rc;
    char                eimerr[250];
    EimRC             * err;
    EimHandle           handle;
    EimPolicyFilterInfo filterInfo;

    /* Set up error structure.                  */
    memset(eimerr,0x00,250);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 250;
.
.
.
    /* Set up policy information                */
    filterInfo.type = EIM_CERTIFICATE_FILTER;
    filterInfo.filter.certFilter.sourceRegistry = "MySourceRegistry";
    filterInfo.filter.certFilter.filterValue = NULL;

    /* Remove the policy filter                 */
    rc = eimRemovePolicyFilter(&handle, &filterInfo, err);
.
.
.
```

eimRemoveRegistry

Purpose

Removes a currently participating registry from the EIM domain.

Notes:

1. You cannot remove a system registry if there are any application registries that are a subset of the system registry.
2. When a registry is removed, an attempt is made to remove all associations for the registry. For policy associations, this includes all policy associations where this registry is either the source registry or the target registry. If there are any policy filters defined for the registry, the policy filters are removed along with any associations to the policy filters.

Format

```
#include <eim.h>


int eimRemoveRegistry(EimHandle      * eim,
                      char           * registryName,
                      EimRC          * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns. A valid connection is required.

registryName
(Input) The name of the registry to remove. Registry names are case-independent (meaning, not case-sensitive).

The following special characters are not allowed in registry names:

```
, = + < > # ; \ *
```

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimAddApplicationRegistry” on page 170
- “eimAddSystemRegistry” on page 190
- “eimChangeRegistry” on page 203
- “eimListRegistries” on page 317

Authorization

EIM data

EIM access groups control access to EIM data. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM administrator

z/OS authorization

No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
EBADDATA	eimrc is not valid.	
EBADNAME	Registry not found or insufficient access to EIM data.	
	EIMERR_NOREG (28)	EIM registry not found or insufficient access to EIM data.
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check the API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use either the eimConnect or eimConnectToMaster API and try the request again.
ENOTSAFE	Cannot delete a system registry when an application registry has this system registry defined.	
	EIMERR_REG_NOTEMPTY (40)	Cannot delete a system registry when there is an application registry defined for this system registry.
EROFS	LDAP connection is for read-only. Need to connect to master. Use the eimConnectToMaster API to get a write connection or use the URL for the master EIM domain controller which is writeable.	
	EIMERR_READ_ONLY (36)	This LDAP connection has ″read-only″ access. A connection to the master LDAP server with read/write is required to complete this operation. Use eimConnectToMaster to get a write connection.

Return Value	Meaning	
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	Unexpected LDAP error.
	EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example illustrates removing an EIM registry:

```
#include <eim.h>

   .
 .
 .
   int           rc;
   char          eimerr[200];
   EimRC       * err;
   EimHandle     handle;

   /* Set up error structure.                  */
   memset(eimerr,0x00,200);
   err = (EimRC *)eimerr;
   err->memoryProvidedByCaller = 200;
 .
 .
 .
   /* Remove the registry                      */
   rc = eimRemoveRegistry(&handle, "MyRegistry", err);
 .
 .
 .
```

eimRetrieveConfiguration

Purpose

This API retrieves the EIM configuration information for the specified or configured profile. Although similar to other @server platform implemenations, the z/OS implementation does differ is the number of parameters passed to the API, and where and how configuration information is stored and retrieved.

Format

```
#include <eim.h>


int eimRetrieveConfiguration(unsigned int     lengthOfEimConfig,
                             EimConfig      * configData,
                             char           * profile,
                             char           * userIdentity,
                             int              ccsid,
                             EimRC          * eimrc)
```

Parameters

lengthOfEimConfig
(Input) The number of bytes the caller provides for the configuration information. The minimum size required is 36 bytes.

configData
(Output) A pointer to the data to return. The EimConfig structure contains information about the returned data. The API returns as much data as space has been provided. The EimConfig structure follows:

```
typedef struct EimConfig
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API.                      */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API.                         */
    int          enable;            /* Flag to indicate if enabled to
                                       participate in EIM domain
                                       0 = not enabled
                                       1 = enabled                      */
    EimListData  ldapURL;           /* ldap URL for domain controller */
    EimListData  localRegistry;     /* Local system registry          */
    EimListData  kerberosRegistry;  /* Kerberos registry              */
    EimListData  x509Registry;      /* X.509 registry                 */
    EimListData  profileName;       /* The name of the profile
                                       storing the ldapURL and
                                       connect info                   */
    EimListData  profClass;         /* The class of the profile       */
    EimListData  profBindDn;        /* The configured bind dn         */
    EimListData  profBindPw;        /* The configured bind password   */
} EimConfig;
```

Note: The actual value of the bind password is not returned by this API. If the bind password is not set in the profile, the length of the profBindPw returned will be zero. If set, the value returned will be YES.

The EimListData structure follows:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                 */
```

```
        unsigned int disp;              /* Displacement to data.  This byte
                                           offset is relative to the start of
                                           the parent structure, i.e. the
                                           structure containing this
                                           structure.                       */
    } EimListData;
```

profile
(Input) The name of a profile containing EIM configuration information for z/OS. The maximum size for this name is 246 bytes. Possible values are:

NULL

a profile name
The unique name of a profile in the LDAPBIND class, the IRR.EIM.DEFAULTS profile in the LDAPBIND class, or the IRR.PROXY.DEFAULTS profile in the FACILITY class.

When profile and userIdentity are both NULL, the API uses the search order for finding EIM configuration information on z/OS. The search order is:

- The LDAPBIND class profile associated with the caller's user ID
- The IRR.EIM.DEFAULTS LDAPBIND class profile
- The IRR.PROXY.DEFAULTS FACILITY class profile

Otherwise, the information is retrieved from the specified profile. Note that if userIdentity is specified at the same time as the profile parameter, an error is returned since you can specify either profile or userIdentity, neither, but not both.

userIdentity
(Input) The user identity with an associated LDAPBIND class profile. The maximum size of this user ID is 8 bytes. Possible values are:

NULL

a user identity
The unique name of a user identity associated with an LDAPBIND class profile.

When profile and userIdentity are both NULL, the API uses the search order for finding EIM configuration information on z/OS. The search order is:

- The LDAPBIND class profile associated with the caller's user identity
- The IRR.EIM.DEFAULTS LDAPBIND class profile
- The IRR.PROXY.DEFAULTS FACILITY class profile

Otherwise, the domain and bind credentials are retrieved from the LDAPBIND class profile associated with the user identity.

The profile parameter must be NULL when userIdentity is used.

ccsid
(Input) The coded character set identifier (CCSID) for the output data. This parameter is ignored.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see "EimRC -- EIM return code parameter for C/C++" on page 164.

Related Information

See the following:

- "eimSetConfiguration" on page 385

Authorization

z/OS authorization

The user identity associated with the application must have one of the following RACF authorities:

- SPECIAL
- CLAUTH authority to the FACILITY and LDAPBIND classes with field-level access checking set up for the fields in the PROXY and EIM segments of both classes.

See the *RACF System Administrator's Guide* for details on how to grant user identities these authorities.

The calling application can be running in system key or supervisor state or one of the following:

- The RACF user ID of the caller's address space has READ authority to the BPX.SERVER profile in the FACILITY class
- The current RACF user ID has READ authority to the IRR.RGETINFO.EIM profile in the FACILITY class

The FACILITY class must be active and RACLISTed before unauthorized (problem program state and keys) will be granted the authority to use this SAF service.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
EBADDATA	eimrc is not valid.	
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value) Error occurred when converting data between code pages.
EINVAL	Input parameter was not valid.	
	EIMERR_CCSID_INVAL (8)	(z/OS does not return this value) CCSID is outside of valid range or CCSID is not supported.
	EIMERR_CONFIG_SIZE (10)	Length of EimConfig is not valid.
	EIMERR_PARM_REQ (34)	Missing required parameter. Please check API documentation.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value) Pointer parameter is not valid.
	EIMERR_SPACE (41)	Unexpected error accessing parameter.
	EIMERR_USERID_INVAL(6019)	User identity does not exist.

Return Value	Meaning
EMVSSAFEXTRERR	SAF/RACF Extract error.
	EIMERR_ZOS_USER_XTR (6002) RACROUTE REQUEST=EXTRACT error retrieving EIM configuration from the caller's USER profile.
	EIMERR_ZOS_XTR_EIM (6003) RACROUTE REQUEST=EXTRACT error retrieving EIM information from a RACF profile.
	EIMERR_ZOS_XTR_PROXY (6005) RACROUTE REQUEST=EXTRACT error retrieving PROXY information from a RACF profile.
	EIMERR_ZOS_R_DCEKEY (6008) R_DCEKEY callable service failed.
	EIMERR_ZOS_R_DCEKEY_BINDPW (6009) R_DCEKEY callable service failed. Bind password is missing.
EMVSSAF2ERR	SAF/RACF error.
	EIMERR_ZOS_XTR_DOMAINDN (6004) EIM domain distinguished name is missing.
	EIMERR_ZOS_XTR_LDAPHOST (6006) PROXY LDAP host is missing.
	EIMERR_ZOS_XTR_BINDDN (6007) PROXY LDAP bind distinguished name is missing.
	EIMERR_ZOS_NO_ACEE (6010) No task or address space ACEE found.
ENAMETOOLONG	A parameter's value is too long.
	EIMERR_PROFILE_SIZE (6016) The profile is too long.
	EIMERR_USERID_SIZE (6017) The user identity is too long.
ENOMEM	Unable to allocate required space.
	EIMERR_NOMEM (27) No memory available. Unable to allocate required space.
EUNKNOWN	Unexpected exception.
	EIMERR_LDAP_ERR (23) Unexpected LDAP error.
	EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Example

The following example retrieves the configuration information and prints out the results.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <string.h>
void printErr(EimRC *err);

void printListData(char * fieldName,
                   void * entry,
                   int offset);

int main (int argc, char *argv[])
{
    int           rc;
    char          eimerr[250];
```

```
EimRC       * err;
char          listData[4000];
EimConfig   * list = (EimConfig *)listData;
EimListData * pwData;
int           pwLen;

/* Set up error structure.                  */
memset(eimerr,0x00,250);
err = (EimRC *)eimerr;
err->memoryProvidedByCaller = 250;

/*
 * Get configuration information using the default
 * search order.
 * (NULL userIdentity and profile parameter)
 */
if (0 != (rc = eimRetrieveConfiguration(4000,
                                        list,
                                        NULL,
                                        NULL,
                                        0,
                                        err)))
{
  /* Return RC and MSID to pinpoint any errors. */
     printf("Retrieve configuration RC = %d, MSID = %d\n",
     rc, err->messageCatalogMessageID);
     printErr(err);
     return -1;
}

/* Print the results                        */
printf("___________\n");
printf("   bytesReturned    = %d\n", list->bytesReturned);
printf("   bytesAvailable   = %d\n", list->bytesAvailable);
printf("\n");
if (0 == list->enable)
    printf("Profile is Disabled.\n");
else
    printf("Profile is Enabled.\n");

printListData("Profile Name",
              list,
              offsetof(EimConfig, profileName));
printListData("Profile Class",
              list,
              offsetof(EimConfig, profClass));
printListData("ldap URL",
              list,
              offsetof(EimConfig, ldapURL));
printListData("local Registry",
              list,
              offsetof(EimConfig, localRegistry));
printListData("kerberos registry",
              list,
              offsetof(EimConfig, kerberosRegistry));
printListData("x.509 registry",
              list,
              offsetof(EimConfig, x509Registry));
printListData("Bind distinguished name",
              list,
              offsetof(EimConfig, profBindDn));

pwData = (EimListData *)((char *)list + offsetof(EimConfig, profBindPw));
pwLen  = pwData->length;
if (pwLen > 0) {
   printf("     The Bind password is set\n");
} else {
```

```
       printf("     The Bind password is NOT set\n");
    }

    return 0;
}

void printListData(char * fieldName,
                   void * entry,
                   int offset)
{
    EimListData * listData;
    char        * data;
    int           dataLength;

    printf("     %s = ",fieldName);

    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}

/* Prints out the error message associated with the EimRC err structure. */

     void printErr(EimRC *err)
     {
         char * msg = NULL;

         msg = eimErr2String(err);

         printf("   ldaperr = %d\n", err->ldapError);
         printf("   msg     = \"%s\"\n",msg);

         free(msg);

         return;
     }
```

eimSetAttribute

Purpose

Sets attributes in the EIM handle structure.

Format

```
#include <eim.h>


int eimSetAttribute(EimHandle          * eim,
                    enum EimHandleAttr   attrName,
                    void               * attrValue,
                    EimRC              * eimrc)
```

Parameters

eim
(Input) The EIM handle that a previous call to eimCreateHandle returns.

attrName
(Input) The name of the attribute to set. This can be:

EIM_HANDLE_CCSID (0) (z/OS does not support this value.) This is the CCSID of character data that the caller of the EIM APIs passes by using the specified EIM handle. This field is a 4-byte integer. When a handle is created, this is set to the job default CCSID.

attrValue
(Input) A pointer to the attribute value.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimConnect” on page 215
- “eimConnectToMaster” on page 220
- “eimCreateHandle” on page 230
- “eimDestroyHandle” on page 239
- “eimGetAttribute” on page 261

Authorization

z/OS authorization
No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_ACCESS (1)	Insufficient access to EIM data.
ENOTSUP	Attribute type is not supported.	
	EIMERR_ATTR_NOTSUPP (6)	The specified attribute is not supported.

eimSetConfiguration

Purpose

This API sets configuration information for use with EIM. On z/OS, this API is not supported and simply returns an error indicating so. To set up EIM configuration information on z/OS, user either RACF commands or callable services or the eimSetConfigurationExt API.

Format

```
#include <eim.h>


int eimSetConfiguration(int             enable,
                        char          * ldapURL,
                        char          * localRegistry,
                        char          * kerberosRegistry,
                        int             ccsid,
                        EimRC         * eimrc)
```

Parameters

enable
(Input) Indicates if this system is able to establish new connections in order to participate in an EIM domain. Possible values are:

0 Not enabled to participate in EIM domain. You cannot establish new connections with the configured EIM domain.

non-zero Enabled to participate in EIM domain. You can establish new connections with the EIM domain.

ldapURL
(Input) A uniform resource locator (URL) that contains the EIM configuration information for the EIM domain controller. This information is used for all EIM operations. The maximum size for this URL is 1000 bytes. Possible values are:

NULL A value of NULL indicates that the LDAP URL that the system stores should not change.

EIM_CONFIG_NONE (*NONE) This value indicates that this system is not configured for EIM.

ldapURL A URL that contains EIM domain controller information. This URL has one of the following formats:

```
ldap://host:port/dn
```

host:port
Is the name of the host on which the EIM domain controller is running. (The port number is optional.)

dn Is the distinguished name for the domain entry.

Examples:

```
ldap://systemx:389/ibm-eimDomainName=myEimDomain,o=myCompany,c=us
ldaps://systemy:636/ibm-eimDomainName=thisEimDomain,o=myCompany,c=us
```

Note: In contrast with `ldap`, `ldaps` indicates that this host and port combination uses SSL and TLS.

localRegistry
(Input) The local EIM system registry name. The maximum size for this registry name is 256 bytes. Possible values are:

NULL	A value of NULL indicates that the local registry name that the system stores should not change.
EIM_CONFIG_NONE	(*NONE) This value indicates that there is no local system registry.
registry	The local EIM system registry name.

kerberosRegistry
(Input) The EIM Kerberos registry name. The maximum size for this registry name is 256 bytes. Possible values are:

NULL	A value of NULL indicates that the EIM Kerberos registry that the system stores should not change.
EIM_CONFIG_NONE	(*NONE) This value indicates that there is no Kerberos registry for EIM.
registry	The EIM Kerberos registry name. This is the Kerberos realm name.

ccsid
(Input) The CCSID of the input data. If the *ccsid* is 0 or 65535, EIM uses the default job CCSID.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

None.

Authorization

No special authorization is needed.

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning
EACCES	Access denied. Not enough permissions to access data.
ENOTSUP	Operation is not supported. **EIMERR_API_NOTSUPP (6012)** The EIM API not supported.

eimSetConfigurationExt

Purpose

Sets the configuration information for use by the system. This information is stored in profiles and retrieved by other EIM APIs, allowing an application to manage the default domain, bind credentials, and registry names for use by the system, servers, or administrative users from an EIM administration application instead of using SAF services and TSO commands. If the profiles don't exist they will be created and can be deleted if requested.

With this API, you can do the following:

- Create, update, or delete a profile that represents the default domain and bind credentials for the system. You can use the following profiles:
  - the IRR.EIM.DEFAULTS profile in the LDAPBIND class. This is the preferred profile to use.
  - the IRR.PROXY.DEFAULTS profile in the FACILITY class. If you are already using policy director and the EIM domain is stored in the same LDAP directory, then this profile can be used when policy director and EIM use the same bind information.
- Create, update, delete, or list a profile that represents the default domain and bind credentials used by a server or EIM administrator. The profile is in the LDAPBIND class. The API also allows you to associate the profile with the server's or administrator's user ID.
- Store the registry names used in the EIM domain for the local registry, the kerberos registry, or the x.509 registry in the IRR.PROXY.DEFAULTS profile. The registry names are not used by EIM applications until they are brought into storage.

Format

```
#include <eim.h>
int eimSetConfigurationExt(EimConfigInfo  * configInfo,
                           EimRC          * eimrc)
```

Parameters

configInfo

The configuration information to be set. The structure layout follows:

```
enum EimConfigFormat {
      EIM_CONFIG_FORMAT_0             /* Information is in configuration
                                       format 0.                      */
  };

  typedef struct EimConfigFormat0
  {
       char * ldapURL;                /* URL for EIM domain controller. */
       char * localRegistry;          /* Local system registry name.    */
       char * kerberosRegistry;       /* Kerberos registry name.        */
       char * x509Registry;           /* X.509 registry name.           */
       char * profile;                /* Profile with EIM config info   */
       char * userIdentity;           /* User id                        */
       char * bindDn;                 /* Bind distinguished name */
       char * bindPw;                 /* Bind password */
              } EimConfigFormat0;
  typedef struct EimConfigInfo
  {
       enum EimConfigFormat format;     /* Format of the config info. */
```

```
    int                     enable; /* Indicate if able to establish
                                       new connections in order to
                                       participate in EIM domain
                                       0 = not enabled
                                       1 = enabled */
    int                     ccsid;  /* CCSID of input data. If 0 or
                                       65535, default job CCSID will
                                       be used. */
    enum EimConfigFunc function;   /* Add/Mod or Delete profile */

    union {
         EimConfigFormat0   format0;
    } config;                      /* Configuration information      */
} EimConfigInfo;
```

Details of the configuration information:

ldapURL
A uniform resource locator (URL) that contains the EIM configuration information for the EIM domain controller. This information will be used for all EIM operations. The maximum size for this URL on z/OS is 1024 bytes.

Possible values are:

NULL Indicates a status that will not change.

EIM_CONFIG_NONE
(*NONE) This value indicates that this system is not configured for EIM.

ldapURL
A URL that contains EIM domain controller information.

The ldapURL has the following format: `ldap://host:port/dn` or `ldaps://host:port/dn`. Where:

host:port
The name of the host on which the EIM domain controller is running with an optional port number.

dn The distinguished name for the domain entry.

ldaps Indicates that this host/port combination uses SSL and TLS.

Examples:

```
ldap://systemx:389/ibm-eimDomainName=myEimDomain,o=myCompany,c=us
ldaps://systemy:636/ibm-eimDomainName=thisEimDomain
```

The URL is broken down into components and stored in the EIM profile. Specifying ***NONE** for this parameter means data in the LDAPHOST field of the PROXY segment and the DOMAINDN field in the EIM segment will be deleted. If the function specified is delete EIM profile (1), this parameter is ignored.

localRegistry
The local EIM system registry name. The maximum size for this name varies by platform, but is 255 bytes on z/OS. The possible values are:

NULL Indicates a value which should not change.

EIM_CONFIG_NONE
(*NONE) This value indicates that there is no local system registry.

registry
The local EIM system registry name.

The registry name is always stored in the EIM segment of the IRR.PROXY.DEFAULTS profile in the FACILITY class. This parameter is ignored when the function is delete EIM profile (1).

kerberosRegistry
The EIM Kerberos registry name. The maximum size for this name is 256 bytes. The possible values are:

NULL A value which should not change.

EIM_CONFIG_NONE
(*NONE) This value indicates that there is no kerberos registry for EIM.

registry
The EIM Kerberos registry name.

The registry name is always stored in the EIM segment of the IRR.PROXY.DEFAULTS profile in the FACILITY class. This parameter is ignored when the function is delete EIM profile (1).

x509Registry
The EIM X.509 registry name. The maximum size for this name varies by platform but is 255 bytes for z/OS. The possible values are:

NULL A value which should not change.

EIM_CONFIG_NONE
(*NONE) This value indicates that there indicates that there is no X.509 registry for EIM..

registry
The EIM X.509 registry name. This will be used when adding source associations for user certificates to the EIM identifier.

The registry name is always stored in the EIM segment of the IRR.PROXY.DEFAULTS profile in the FACILITY class. This parameter is ignored when the function is delete EIM profile (1).

function
The action to be performed on the profile. Possible values are:

EIM_CONFIG_FUNC_ADDMOD(0)
Create or update the profile containing EIM configuration data.

EIM_CONFIG_FUNC_DEL(1)
Delete the profile containing EIM configuration data.

profile
The name of a profile containing EIM configuration data. The maximum size is 246 bytes. Possible values are:

NULL A value which should not change.

EIM_CONFIG_NONE
(*NONE) This value indicates that association of the LDAPBIND class profile with the userIdentity should be broken.

system default profile
IRR.EIM.DEFAULTS profile in the LDAPBIND class or IRR.PROXY.DEFAULTS profile in the FACILITY class

server or administrative user's profile
a profile in the LDAPBIND class

Notes:

1. If a profile name is specified without a user identity, the profile is created and updated as requested.
2. When a profile name is specified wth a user identity and the profile is something other than IRR.EIM.DEFAULTS or IRR.PROXY.DEFAULTS, the profile is created if it doesn't exist, and associated the specified user identity.
3. When EIM_CONFIG_NONE (*NONE) is used for the profile name and a user identity is specified, the association the user identity may have with an LDAPBIND class profile is broken. All other parameters are ignored in this case.

userIdentity
The user ID associated with an LDAPBIND class profile. The maximum size for this user ID is 8 bytes. Possible values are:

NULL Indicates a value which will not change.

user identity
The specific user identity you wish to associate with the LDAPBIND class profile. Possible values are:

- NULL
- a user identity

bindDn
The bind distinguished name that is stored in the profile. The bind distinguished name is the identity used when binding with LDAP. This parameter allows the information to be added, replaced, or removed from a profile. This parameter is ignored when the function is delete EIM profile (1). The maximum size for a bindDn varies by platform but is 1023 bytes.

The bindDn parameter may have one of the following values:

NULL A value of NULL indicates that it should not change.

EIM_CONFIG_NONE
(*NONE) This value indicates that there is no bindDn.

value The bindDn. This value is used when establishing a connection with the domain name stored in the profile.

bindPw
The bind password that is stored in the profile. The bind password is used when binding with LDAP. This parameter allows the information to be added, replaced, or removed from a profile. This parameter is ignored when the function is delete EIM profile (1). The maximum size for a bindPn varies by platform but is 128 bytes for z/OS.

The bindPw parameter may have one of the following values:

NULL A value of NULL indicates that it should not change.

EIM_CONFIG_NONE
(*NONE) This value indicates that there is no bindPw.

value The bindPw. This value is used when establishing a connection with the domain name stored in the profile.

This parameter is ignored when the profile name is IRR.EIM.DEFAULTS or IRR.PROXY.DEFAULTS.

enable
(Input) Indicates if this system is able to establish new connections in order to participate in an EIM domain. Possible values are:

0 Not enabled to participate in EIM domain. New connections cannot be established with the configured EIM domain.

non-zero
Enabled to participate in EIM domain. New connections may be established with the EIM domain.

This parameter is ignored when the function is delete EIM profile (1).

ccsid
(Input) The CCSID of the input data. If the ccsid is 0 or 65535 the default job ccsid will be used. The CCSID parameter is not used by all platforms. See the authorization section for specific details.

eimrc
(Input/Output) The structure in which to return error code information. If the return value is not 0, EIM sets eimrc with additional information. This parameter can be NULL. For the format of the structure, see “EimRC -- EIM return code parameter for C/C++” on page 164.

Related Information

See the following:

- “eimRetrieveConfiguration” on page 377

Authorization

EIM data
This API does not connect to the LDAP Server, so there is no EIM data being accessed. Therefore, no EIM authority is needed.

z/OS authorization
The RACF user ID of the caller must satisfy the following requirements:

The calling application can be running in system key or supervisor state or one of the following:

- The RACF user ID of the caller’s address space has READ authority to the BPX.SERVER profile in the FACILITY class
- The current RACF user ID has READ authority to the IRR.RGETINFO.EIM profile in the FACILITY class

For applications that are not authorized (problem program state and keys), the current RACF user ID must satisfy the following requirements:

- Have READ authority to the following profiles in the FACILITY class:
  - IRR.RADMIN.ALTUSER
  - IRR.RADMIN.RDEFINE
  - IRR.RADMIN.RALTER
  - IRR.RADMIN.RDELETE
- Have authority to issue the following commands:
  - ALTUSER
  - RALTER

- RDEFINE
- RDELETE

The FACILITY class must be active and RACLISTed before the application will be granted authority to use this SAF service

Return Values

The following table lists the return values from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

Return Value	Meaning	
0	Request was successful.	
EACCES	Access denied. Not enough permissions to access data.	
	EIMERR_AUTH_ERR (7)	Insufficient authority for the operation.
EBADDATA	eimrc is not valid.	
EBUSY	Unable to allocate internal system object.	
	EIMERR_NOLOCK (26)	(z/OS does not return this value.) Unable to allocate internal system object.
ECONVERT	Data conversion error.	
	EIMERR_DATA_CONVERSION (13)	(z/OS does not return this value.) Error occurred when converting data between code pages.

Return Value	Meaning	
EINVAL	Input parameter was not valid.	
	EIMERR_CCSID_INVAL (8)	(z/OS does not return this value.) CCSID is outside of valid range or CCSID is not supported.
	EIMERR_CHAR_INVAL (21)	A restricted character was used in the object name. Check the API for a list of restricted characters.
	EIMERR_PROTECT_INVAL (22)	The protect parameter in EimSimpleConnectInfo is not valid.
	EIMERR_PARM_REQ (34)	At a minimum one registry name or a profile name must be specified.
	EIMERR_PTR_INVAL (35)	(z/OS does not return this value.) Pointer parameter is not valid.
	EIMERR_URL_NODN (45)	URL has no Distinguished Name, and is required.
	EIMERR_URL_NODOMAIN (46)	URL has no domain and is required.
	EIMERR_URL_NOHOST (47)	URL does not have a host.
	EIMERR_URL_NOTLDAP (49)	URL does not begin with ldap.
	EIMERR_CONN_INVAL (54)	The connectinfo type is not valid.
	EIMERR_INVALID_DN (66)	Distinguished Name (DN) is not valid.
	EIMERR_CONFIG_FORMAT_INVAL (68)	Configuration format is not valid
	EIMERR_ZOS_FUNCTION_INVAL (6014)	The value specified for the function parameter is not valid.
	EIMERR_USERID_MUST_BE_NULL(6015)	The user identity parameter must be NULL when specified with IRR.EIM.DEFAULTS or IRR.PROXY.DEFAULTS profiles names.
	EIMERR_FUNCTION_INVAL (6016)	The function parameter is not valid.
	EIMERR_USERID_INVAL(6019)	User identity does not exist.
EMVSSAF2ERR	SAF/RACF error.	
	EIMERR_ZOS_NO_ACEE (6010)	No task or address space ACEE found.
ENAMETOOLONG	A parameter's value is too long..	
	EIMERR_REGNAME_SIZE (39)	Registry name is too large.
	EIMERR_URL_SIZE (51)	Configuration URL is too large.
	EIMERR_PROFILE_SIZE (6016)	The profile is too long.
	EIMERR_USERID_SIZE (6017)	The user identity is too long.
	EIMERR_BINDPW_SIZE (6019)	The bind password is too long.
	EIMERR_BINDDN_SIZE (6020)	The bind Distinguished Name (DN) is too long.
ENOMEM	Unable to allocate required space.	
	EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.

Return Value	Meaning	
EUNKNOWN	Unexpected exception.	
	EIMERR_LDAP_ERR (23)	(z/OS does not return this value.) Unexpected LDAP error.
	EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Example

The following example sets the configuration information but it is not enabled.

```
#include <eim.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>

int main (int argc, char *argv[])
{
    int             rc;
    char            ldapURL[250];
    char            eimerr[1000];
    EimRC         * err;
    EimConfigInfo   configInfo;
    char          * errstr;

    err = (EimRC *)eimerr;
    memset(eimerr, 0x00, 1000);
    err->memoryProvidedByCaller = 1000;

    /*-------------------------------------------------------------------*/
    /*                                                                   */
    /*                                                                   */
    /*                                                                   */
    /*                                                                   */
    /*-------------------------------------------------------------------*/
    memset(&configInfo, 0x00, sizeof(EimConfigInfo));
    configInfo.format = EIM_CONFIG_FORMAT_0;
    configInfo.enable = 1;
    configInfo.function = EIM_CONFIG_FUNC_ADDMOD;
    configInfo.config.format0.ldapURL =
       "ldap://localhost:389/ibm-eimDomainName=MyDomain,o=MyOrg,c=us";
    configInfo.config.format0.profile = "timmys.ldapbind.profile";
    configInfo.config.format0.userIdentity = "timmy";
    configInfo.config.format0.localRegistry = "SYS1.RACFDB";
    configInfo.config.format0.kerberosRegistry = "z/OS KDC";
    configInfo.config.format0.x509Registry = "PKI Services";
    configInfo.config.format0.bindDn = "cn=racfid=timmy,profiletype=user,o=racfdb,c=us";
    configInfo.config.format0.bindPw = "secret";

    if (eimSetConfigurationExt(&configInfo, err)) {
       if (NULL == (errstr = eimErr2String(err))) {
          printf("eimSetConfigurationExt failed; rc=(%d) msgid(%d) errno: %s\n"
                , err->returnCode, err->messageCatalogMessageID
                , strerror(err->returnCode));
       } else {
          printf("eimSetConfigurationExt failed; rc=(%d) msgid(%d): %s\n"
                , err->returnCode, err->messageCatalogMessageID, errstr);
          free(errstr);
       }
    }

    return 0;
}
```

Chapter 12. EIM header file and example

eim.h

The eim.h header file resides in the HFS in the /usr/include directory. You include eim.h in all applications using EIM APIs.

Note: For the latest version of the eim.h header file refer to the Hierarchical File System (HFS).

```
#ifdef __COMPILER_VER__
#pragma filetag ("IBM-1047")
#endif
/*
 * Source file: eim.h 1.13
 * Last Updated: 9/11/03 14:24:51
 */
/*****************************************************************/
/*                                                               */
/*    Licensed Materials - Property of IBM                       */
/*    5694-A01                                                   */
/*    (C) Copyright IBM Corp. 2002, 2004                         */
/*    Status = HIT7709                                           */
/*                                                               */
/*****************************************************************/
#ifndef EIM_h
#define EIM_h
#ifdef __cplusplus
   #pragma info(none)
#else
   #pragma nomargins nosequence
   #pragma checkout(suspend)
#endif
/*** START HEADER FILE SPECIFICATIONS ******************************/
/*                                                               */
/* Header File Name:  eim.h                                      */
/*                                                               */
/* Descriptive Name: Enterprise Identity Mapping (EIM) APIs      */
/*                                                               */
/* Description:                                                  */
/*                                                               */
/*          Defines prototypes, macros, variables, and           */
/*          structures to be used with the EIM APIs.             */
/*                                                               */
/* Header Files Included:                                        */
/*                                                               */
/*                                                               */
/* Macros List:                                                  */
/*                                                               */
/*                                                               */
/* Structure List:                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/* Function Prototype List:                                      */
/*                                                               */
/*                                                               */
/* Change Activity:                                              */
/*                                                               */
/*   CFD List:                                                   */
/*                                                               */
/*   FLAG REASON       LEVEL DATE    PGMR     CHANGE DESCRIPTION     */
/*   ---- ------------ ----- ------- -------- ---------------------- */
/*   $A0= D9860600     5D20  020202  ROCH     New include.           */
/*   $A1= P9A04903     5D20  020330  ROCH     Fix AIX registry type. */
```

```
/*   $L1= EIM        HIT7708 091901  RDC1     EIM                    */
/*   $P1= MG01014    HIT7708 062402  $PDTCG1: krb/ssl removal    @P1A*/
/*   $P2= MG01149    HIT7708 081502  $PDTCG1: Define errnos      @P2A*/
/*   $L2= MG01076    HIT7708 101002  $PDTCG1: krb/ssl bind support   */
/*   $01= D9121900    V5R3MO 021215  ROCH     Add policy support.    */
/*                                            Add message catalog    */
/*                                            Id numbers.            */
/*   $02= D9121908    V5R3MO 021215  ROCH     Add new config.        */
/*                                            Add assoc type to      */
/*                                            EimIdentifier struct   */
/*   $03= D9121910    V5R3MO 030216  ROCH     Reorganize structures. */
/*   $L3= EIME       hit7709 030325  $PDTCG:  Added new constant     */
/*   $04= P9A26515    V5R3MO 030330  ROCH     Add linux registry type*/
/*   $L4= EIME       hit7709 030506  $PDTCG:  Added Config support   */
/*   $05= P9A28411    V5R3MO 030518  ROCH     Add Tivoli Access      */
/*                                            Manager type, duplicate*/
/*                                            of Policy Director     */
/*   $06= P9A28415    V5R3MO 030720  ROCH     Add Domino reg types.  */
/*   $07= P9A35259    V5R3MO 030803  ROCH     Add new Windows reg    */
/*                                            type defines.          */
/*   $08= P9A36692    V5R3MO 030831  ROCH     Add eimGetVersion      */
/*** END HEADER FILE SPECIFICATIONS *********************************/


#if (__OS400_TGTVRM__>=510)
#pragma datamodel(P128)
#endif

#ifndef __MVS__                                                 /*@P1C*/
#include "gssapi.h"
#else                                                           /*@P1C*/
#include <skrb/gssapi.h>                                        /*@P1A*/
#endif                                                          /*@P1A*/

#ifdef __cplusplus
extern "C" {
#endif

#pragma enum(4)
/*--------------------------------------------------------------------*/
/* On z/OS, define non-standard errno values if not defined in the    */
/* errno.h file                                                 @P2A*/
/*--------------------------------------------------------------------*/
#ifdef __MVS__                                                  /*@P2A*/
 #include <errno.h>                                             /*@P2A*/
 #ifndef  EBADDATA                                              /*@P2A*/
  #define  EBADDATA          245  /* Data invalid.               @P2A*/
 #endif                                                         /*@P2A*/
 #ifndef  EUNKNOWN                                              /*@P2A*/
  #define  EUNKNOWN          246  /* Unknown system state.       @P2A*/
 #endif                                                         /*@P2A*/
 #ifndef  ENOTSUP                                               /*@P2A*/
  #define  ENOTSUP           247  /* Operation not supported.    @P2A*/
 #endif                                                         /*@P2A*/
 #ifndef  EBADNAME                                              /*@P2A*/
  #define  EBADNAME          248  /* Invalid file name specified.@P2A*/
 #endif                                                         /*@P2A*/
 #ifndef  ENOTSAFE                                              /*@P2A*/
  #define  ENOTSAFE          249  /* Function not allowed.       @P2A*/
 #endif
#endif

/*--------------------------------------------------------------------*/
/*    Constants                                                       */
/*--------------------------------------------------------------------*/

#define EIM_HANDLE_SIZE            16 /* EIM Handle size             */
```

```
#define EIM_LIST_MIN_SIZE          20 /* Minimal size for EimList
                                         structure                    */
#define EIM_RC_MIN_SIZE            48 /* Minimal size for EimRc
                                         structure                    */
#define EIM_CONFIG_MIN_SIZE        36 /* Minimal size for EimConfig
                                         structure                    */
#define EIM_ATTRIBUTE_MIN_SIZE     16 /* Minimal size for EimAttribute
                                         structure                    */
#define EIM_USER_IDENTITY_MIN_SIZE 16 /* Minimal size for EimUserIdentity
                                         structure                @L3A*/


#define EIM_LDAP_URL_MAX    1000 /* Maximum size for LDAP URL         */
#define EIM_LOCREG_MAX       256 /* Maximum size for local registry   */
#define EIM_KRBREG_MAX       256 /* Maximum size for kerberos registry*/
#define EIM_X509REG_MAX      256 /* Maximum size for X.509 reg    @01A*/

#define EIM_UNIQUE_ADD_SIZE   20 /* Minimal additional size required for
                                    identifier unique name            */

/*--------------------------------------------------------------------*/
/*    Configuration constants                                         */
/*--------------------------------------------------------------------*/
#define EIM_CONFIG_NONE "*NONE"

/*--------------------------------------------------------------------*/
/*    Normalization methods                                           */
/*--------------------------------------------------------------------*/
#define EIM_NORM_CASE_IGNORE "-caseIgnore"
#define EIM_NORM_CASE_EXACT  "-caseExact"

/*--------------------------------------------------------------------*/
/*    Registry types                                                  */
/*--------------------------------------------------------------------*/
#define EIM_REGTYPE_RACF "1.3.18.0.2.33.1-caseIgnore"
#define EIM_REGTYPE_OS400 "1.3.18.0.2.33.2-caseIgnore"
#define EIM_REGTYPE_KERBEROS_EX "1.3.18.0.2.33.3-caseExact"
#define EIM_REGTYPE_KERBEROS_IG "1.3.18.0.2.33.4-caseIgnore"
#define EIM_REGTYPE_WIN_DOMAIN_KERB_IG "1.3.18.0.2.33.4-caseIgnore"
                                                          /* @07A*/
#define EIM_REGTYPE_AIX      "1.3.18.0.2.33.5-caseExact"
#define EIM_REGTYPE_NDS      "1.3.18.0.2.33.6-caseIgnore"
#define EIM_REGTYPE_LDAP     "1.3.18.0.2.33.7-caseIgnore"
#define EIM_REGTYPE_POLICY_DIRECTOR   "1.3.18.0.2.33.8-caseIgnore"
#define EIM_REGTYPE_TIVOLI_ACCESS_MANAGER "1.3.18.0.2.33.8-caseIgnore"
                                                          /* @05A*/
#define EIM_REGTYPE_WIN2K    "1.3.18.0.2.33.9-caseIgnore"
#define EIM_REGTYPE_WINDOWS_LOCAL_WS "1.3.18.0.2.33.9-caseIgnore"
                                                          /* @07A*/
#define EIM_REGTYPE_X509        "1.3.18.0.2.33.10-caseIgnore"  /* @02C*/
#define EIM_REGTYPE_LINUX       "1.3.18.0.2.33.11-caseIgnore"  /* @04A*/
#define EIM_REGTYPE_DOMINO_LONG  "1.3.18.0.2.33.12-caseIgnore" /* @06A*/
#define EIM_REGTYPE_DOMINO_SHORT "1.3.18.0.2.33.13-caseIgnore" /* @06A*/


/*--------------------------------------------------------------------*/
/*    Registry alias types                                            */
/*--------------------------------------------------------------------*/
#define EIM_ALIASTYPE_DNS       "DNSHostName"
#define EIM_ALIASTYPE_KERBEROS  "KerberosRealm"
#define EIM_ALIASTYPE_ISSUER    "IssuerDN"
#define EIM_ALIASTYPE_ROOT      "RootDN"
#define EIM_ALIASTYPE_TCPIP     "TCPIPAddress"
#define EIM_ALIASTYPE_LDAPDNSHOSTNAME  "LdapDnsHostName"
#define EIM_ALIASTYPE_OTHER     "Other"                   /* @01A*/
```

```
/*-------------------------------------------------------------------*/
/*    EimHandle Attributes                                           */
/*-------------------------------------------------------------------*/
enum EimHandleAttr {
    EIM_HANDLE_CCSID,
    EIM_HANDLE_DOMAIN,              /* Retrieved but not changed      */
    EIM_HANDLE_HOST,                /* Retrieved but not changed      */
    EIM_HANDLE_PORT,                /* Retrieved but not changed      */
    EIM_HANDLE_SECPORT,             /* Retrieved but not changed      */
    EIM_HANDLE_MASTER_HOST,         /* Retrieved but not changed      */
    EIM_HANDLE_MASTER_PORT,         /* Retrieved but not changed      */
    EIM_HANDLE_MASTER_SECPORT       /* Retrieved but not changed      */
};

/*-------------------------------------------------------------------*/
/*    Attributes to change, add, remove, enable, or disable          */
/*-------------------------------------------------------------------*/
enum EimChangeType {
    EIM_CHG,
    EIM_ADD,
    EIM_RMV,
    EIM_ENABLE,                                             /* @01A*/
    EIM_DISABLE                                             /* @01A*/
};

enum EimDomainAttr
{                                   /* Change type:                */
    EIM_DOMAIN_DESCRIPTION,         /*     Change                  */
    EIM_DOMAIN_POLICY_ASSOCIATIONS  /*     Enable/Disable        @01A*/
};


enum EimRegistryAttr
{                                   /* Change type:                */
    EIM_REGISTRY_DESCRIPTION,       /*     Change                  */
    EIM_REGISTRY_LABELEDURI,        /*     Change                  */
    EIM_REGISTRY_MAPPING_LOOKUP,    /*     Enable/Disable        @01A*/
    EIM_REGISTRY_POLICY_ASSOCIATIONS /*    Enable/Disable        @01A*/
};

enum EimRegistryUserAttr
{                                   /* Change type:                */
    EIM_REGISTRYUSER_DESCRIPTION,   /*     Change                  */
    EIM_REGISTRYUSER_ADDL_INFO      /*     Add or remove           */
};

enum EimIdentifierAttr
{                                   /* Change type:                */
    EIM_IDENTIFIER_DESCRIPTION,     /*     Change                  */
    EIM_IDENTIFIER_NAME,            /*     Add or remove           */
    EIM_IDENTIFIER_ADDL_INFO        /*     Add or remove           */
};


/*-------------------------------------------------------------------*/
/*    EIMAssociationType                                             */
/*-------------------------------------------------------------------*/
enum EimAssociationType {
    EIM_ALL_ASSOC,
    EIM_TARGET,
    EIM_SOURCE,
    EIM_SOURCE_AND_TARGET,
    EIM_ADMIN,
    EIM_ALL_POLICY_ASSOC,                                   /* @01A*/
    EIM_CERT_FILTER_POLICY,                                 /* @01A*/
    EIM_DEFAULT_REG_POLICY,                                 /* @01A*/
    EIM_DEFAULT_DOMAIN_POLICY                               /* @01A*/
```

```
};

/*--------------------------------------------------------------------*/
/*    EIMRegistryKind                                                 */
/*--------------------------------------------------------------------*/
enum EimRegistryKind {
    EIM_ALL_REGISTRIES,            /* System and application         */
    EIM_SYSTEM_REGISTRY,
    EIM_APPLICATION_REGISTRY
};

/*--------------------------------------------------------------------*/
/*    EIMHandle                                                       */
/*--------------------------------------------------------------------*/
typedef struct EIMHandle
{
   char        handle[EIM_HANDLE_SIZE];
}  EimHandle;

/*--------------------------------------------------------------------*/
/*    Eim Connect Information                                         */
/*--------------------------------------------------------------------*/
enum EimPasswordProtect {
    EIM_PROTECT_NO,
    EIM_PROTECT_CRAM_MD5,
    EIM_PROTECT_CRAM_MD5_OPTIONAL
};
enum EimConnectType {
    EIM_SIMPLE,
    EIM_KERBEROS,
    EIM_CLIENT_AUTHENTICATION
};

typedef struct EimSimpleConnectInfo
{
     enum EimPasswordProtect protect;
     char * bindDn;
     char * bindPw;
} EimSimpleConnectInfo;

typedef struct EimSSLInfo
{
     char * keyring;
     char * keyring_pw;
     char * certificateLabel;
} EimSSLInfo;

/* NOTE:  for compatability, do not add any information to the union  */
/*        in this structure that will increase the size of the union. */
typedef struct EimConnectInfo
{
     enum EimConnectType type;
     union {
         gss_cred_id_t * kerberos;
         EimSimpleConnectInfo simpleCreds;
     } creds;
     EimSSLInfo * ssl;
} EimConnectInfo;
/*--------------------------------------------------------------------*/
/*    EimIdAction                                                     */
/*--------------------------------------------------------------------*/
enum EimIdAction {
    EIM_FAIL,
    EIM_GEN_UNIQUE
};
/*--------------------------------------------------------------------*/
/*    EimIdentifierInfo                                               */
```

```
/*--------------------------------------------------------------------*/
enum EimIdType {
    EIM_UNIQUE_NAME,
    EIM_ENTRY_UUID,
    EIM_NAME
};

/* NOTE:  for compatability, do not add any information to the union  */
/*        in this structure that will increase the size of the union. */
typedef struct EimIdentifierInfo
{
    union {
        char       * uniqueName;
        char       * entryUUID;
        char       * name;
    } id;
    enum EimIdType        idtype;
} EimIdentifierInfo;

/*--------------------------------------------------------------------*/
/*    EimStatus                                                       */
/*--------------------------------------------------------------------*/
enum EimStatus {
    EIM_STATUS_NOT_ENABLED,
    EIM_STATUS_ENABLED
};                                                          /* @01A*/

/*--------------------------------------------------------------------*/
/*    Eim Policy Information                                          */
/*--------------------------------------------------------------------*/
enum EimPolicyFilterType {
    EIM_ALL_FILTERS,
    EIM_CERTIFICATE_FILTER
};                                                          /* @01A*/
typedef struct EimCertificatePolicyFilter
{
     char * sourceRegistry;
     char * filterValue;
}  EimCertificatePolicyFilter;                              /* @01A*/

typedef struct EimPolicyFilterInfo
{
     enum EimPolicyFilterType type;
     union {
         EimCertificatePolicyFilter   certFilter;
     } filter;
}  EimPolicyFilterInfo;                                     /* @03C*/

typedef struct EimCertPolicyFilterSubsetInfo
{
     char * subjectFilter;
     char * issuerFilter;
}  EimCertPolicyFilterSubsetInfo;                           /* @01A*/

typedef struct EimPolicyFilterSubsetInfo
{
     union {
         EimCertPolicyFilterSubsetInfo   certFilter;
     } subset;
}  EimPolicyFilterSubsetInfo;                               /* @01A*/

typedef struct EimCertificateFilterPolicyAssociation
{
     char * sourceRegistry;
     char * filterValue;
     char * targetRegistry;
     char * targetRegistryUserName;
```

```
}  EimCertificateFilterPolicyAssociation;                      /* @01A*/

typedef struct EimDefaultRegistryPolicyAssociation
{
     char * sourceRegistry;
     char * targetRegistry;
     char * targetRegistryUserName;
}  EimDefaultRegistryPolicyAssociation;                       /* @01A*/

typedef struct EimDefaultDomainPolicyAssociation
{
     char * targetRegistry;
     char * targetRegistryUserName;
}  EimDefaultDomainPolicyAssociation;                         /* @01A*/

typedef struct EimPolicyAssociationInfo
{
     enum EimAssociationType type;
     union {
         EimCertificateFilterPolicyAssociation   certFilter;
         EimDefaultRegistryPolicyAssociation     defaultRegistry;
         EimDefaultDomainPolicyAssociation       defaultDomain;
     } policyAssociation;
}  EimPolicyAssociationInfo;                                  /* @03C*/

/*--------------------------------------------------------------------*/
/*    Eim User Identity Information                                   */
/*--------------------------------------------------------------------*/
enum EimUserIdentityType {
    EIM_DER_CERT,
    EIM_BASE64_CERT,
    EIM_CERT_INFO
};                                                            /* @01A*/

enum EimUserIdentityFormatType {
    EIM_REGISTRY_USER_NAME
};                                                            /* @01A*/

typedef struct EimCertificateInfo
{
     char          * issuerDN;
     char          * subjectDN;
     unsigned char * publicKey;
     unsigned int    publicKeyLen;
}  EimCertificateInfo;                                       /* @O1A*/

typedef struct EimCertificate
{
     char         * certData;
     unsigned int   certLength;
}  EimCertificate;                                           /* @01A*/

typedef struct EimUserIdentityInfo
{
     enum EimUserIdentityType type;
     union {
         EimCertificateInfo   certInfo;
         EimCertificate       cert;
     } userIdentityInfo;
}  EimUserIdentityInfo;                                      /* @03C*/

/*--------------------------------------------------------------------*/
/*    Eim Configuration Information                                   */
/*--------------------------------------------------------------------*/
enum EimConfigFormat {
    EIM_CONFIG_FORMAT_0
};                                                            /* @02A*/
```

```
#ifdef __MVS__                                                  /* @L4A*/
enum EimConfigFunc {                                            /* @L4A*/
    EIM_CONFIG_FUNC_ADDMOD,         /* Add/Modify a profile       @L4A*/
    EIM_CONFIG_FUNC_DEL             /* Delete a profile          @L4A*/
};                                                              /* @L4A*/
#endif                                                          /* @L4A*/

typedef struct EimConfigFormat0
{
     char * ldapURL;
     char * localRegistry;
     char * kerberosRegistry;
     char * x509Registry;
#ifdef __MVS__                                                  /* @L4A*/
     char           * profile;      /* RACF profile name         @L4A*/
     char           * userIdentity; /* userid                    @L4A*/
     char           * bindDn;       /* Bind distinguished name   @L4A*/
     char           * bindPw;       /* Bind password name        @L4A*/
#endif                                                          /* @L4A*/
}  EimConfigFormat0;                                            /* @02A*/

typedef struct EimConfigInfo
{
     enum EimConfigFormat format;
     int                  enable;
     int                  ccsid;
#ifdef __MVS__                                                  /* @L4A*/
     enum EimConfigFunc   function; /* Add/Mod or Delete profile  @L4A*/
#endif                                                          /* @L4A*/
     union {
         EimConfigFormat0   format0;
     } config;
}  EimConfigInfo;                                               /* @03C*/

/*------------------------------------------------------------------*/
/*    Eim Version                                                   */
/*------------------------------------------------------------------*/
enum EimVersion {
    EIM_VERSION_0,   /* EIM is not supported on the specified host.
                                                        @08A*/
    EIM_VERSION_1,   /* EIM version 1 is supported by the specified
        host. This host will support EIM functionality
        provided with the first version of EIM.     @08A*/
    EIM_VERSION_2   /* EIM version 2 is supported by the specified
       host.  This host will support EIM functionality
       provided with the second version of EIM, which
       includes policy association support.         @08A*/
};

/*------------------------------------------------------------------*/
/*    Eim Host information for version                              */
/*------------------------------------------------------------------*/
enum EimHostInfoType {
    EIM_HANDLE,
    EIM_LDAP_URL
};                                                              /* @08A*/

typedef struct EimHostInfo
{
    enum EimHostInfoType  hostType;
    union {
 EimHandle       * eim;
 char            * ldapURL;
    } hostInfo;
} EimHostInfo;                                                  /* @08A*/
```

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/*    Return code structure                                           */
/*                                                                    */
/*--------------------------------------------------------------------*/
typedef struct EimRC {
    unsigned int memoryProvidedByCaller; /* Input: Size of the entire RC
                                     structure. This is filled in by
                                     the caller. This is used to tell
                                     the API how much space was provided
                                     for substitution text           */
    unsigned int  memoryRequiredToReturnData;/* Output: Filled in by API
                                     to tell caller how much data could
                                     have been returned. Caller can then
                                     determine if the caller provided
                                     enough space (i.e. if the entire
                                     substitution string was able to be
                                     copied to this structure.       */
    int  returnCode;                /* Same as the errno returned as the
                                     rc for the API                  */
    int messageCatalogSetNbr;       /* Message catalog set number      */
    int messageCatalogMessageID;    /* Message catalog message id      */
    int ldapError;                  /* ldap error, if available        */
    int sslError;                   /* SLL error, if available         */
    char     reserved[16];          /* Reserved for future use         */
    unsigned int substitutionTextLength; /* Length of substitution text
                                     excluding a null-terminator which
                                     may or may not be present       */
    char substitutionText[1];       /* further info describing the
                                     error.                          */
} EimRC;

/*--------------------------------------------------------------------*/
/*                                                                    */
/*    Access structures                                               */
/*                                                                    */
/*--------------------------------------------------------------------*/
enum EimAccessUserType {
    EIM_ACCESS_DN,
    EIM_ACCESS_KERBEROS,
    EIM_ACCESS_LOCAL_USER
};

typedef struct EimAccessUser
{
    union {
        char * dn;
        char * kerberosPrincipal;
        char * localUser;
    } user;
    enum EimAccessUserType userType;
} EimAccessUser;


enum EimAccessType {
    EIM_ACCESS_ADMIN,
    EIM_ACCESS_REG_ADMIN,
    EIM_ACCESS_REGISTRY,
    EIM_ACCESS_IDENTIFIER_ADMIN,
    EIM_ACCESS_MAPPING_LOOKUP
};
enum EimAccessIndicator {
    EIM_ACCESS_NO,
    EIM_ACCESS_YES
};
/*--------------------------------------------------------------------*/
/*                                                                    */
```

```
/*    EimListData  - this is used to access the data elements.        */
/*    EimSubList   - this is used to access sub lists within the      */
/*              list information returned.                            */
/*--------------------------------------------------------------------*/
typedef struct EimListData
{
    unsigned int length;           /* Length of data                  */
    unsigned int disp;             /* Displacement to data.  This byte
                                      offset is relative to the start
                                      of the parent structure i.e. the
                                      structure containing this
                                      structure                       */
} EimListData;

typedef struct EimSubList
{
    unsigned int listNum;          /* Number of entries in the list  */
    unsigned int disp;             /* Displacement to sublist. This
                                      byte offset is relative to the
                                      start of the parent structure i.e.
                                      the structure containing this
                                      structure                       */
} EimSubList;
/*--------------------------------------------------------------------*/
/*                                                                    */
/*    EimConfig                                                       */
/*        Information returned from eimRetrieveConfiguration() API.   */
/*--------------------------------------------------------------------*/
typedef struct EimConfig
{
    unsigned int bytesReturned;    /* Number of bytes actually returned
                                      by the API                      */
    unsigned int bytesAvailable;   /* Number of bytes of available data
                                      that could have been returned by
                                      the API                         */
    int          enable;           /* Flag to indicate if enabled to
                                      participate in EIM domain
                                      0 = not enabled
                                      1 = enabled                     */
    EimListData  ldapURL;          /* ldap URL for domain controller */
    EimListData  localRegistry;    /* Local system registry          */
    EimListData  kerberosRegistry; /* Kerberos registry              */
    EimListData  x509Registry;     /* X.509 registry            @01A*/
#ifdef __MVS__                                                 /*@L4A*/
    EimListData  profileName;      /* The name of the profile
                                      storing the ldapURL and
                                      connect info                    */
    EimListData  profClass;        /* Class of configured profile    */
    EimListData  profBindDn;       /* The configured bind dn         */
    EimListData  profBindPw;       /* The configured bind password   */
#endif                                                         /*@L4A*/
} EimConfig;
/*--------------------------------------------------------------------*/
/*                                                                    */
/*    EimAttribute                                                    */
/*        Information returned from eimGetAttribute() API.            */
/*--------------------------------------------------------------------*/
typedef struct EimAttribute
{
    unsigned int bytesReturned;    /* Number of bytes actually returned
                                      by the API                      */
    unsigned int bytesAvailable;   /* Number of bytes of available data
                                      that could have been returned by
                                      the API                         */
    EimListData  attribute;        /* handle attribute               */
} EimAttribute;
/*--------------------------------------------------------------------*/
```

```
/*                                                                   */
/*    EimList - this is used by all EIM APIs that return a list.     */
/*              It gives information on the amount of information    */
/*              returned and then gives access to the first list     */
/*              entry.                                               */
/*-------------------------------------------------------------------*/
typedef struct EimList
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API                    */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API                       */
    unsigned int entriesReturned;   /* Number of entries actually
                                       returned by the API           */
    unsigned int entriesAvailable;  /* Number of entries available to be
                                       returned by the API           */
    unsigned int firstEntry;        /* Displacement to the first linked
                                       list entry. This byte offset is
                                       relative to the start of the
                                       EimList structure.            */
} EimList;

/*-------------------------------------------------------------------*/
/*    EimDomain                                                      */
/*       List information returned by the following APIs:            */
/*              eimListDomains                                       */
/*-------------------------------------------------------------------*/
typedef struct EimDomain
{
    unsigned int nextEntry;         /* Displacement to next entry. This
                                       byte offset is relative to the
                                       start of this structure       */
    EimListData name;               /* Domain name                   */
    EimListData dn;                 /* Distinguished name for the domain
                                                                     */
    EimListData description;        /* Description                   */
    enum EimStatus  policyAssociations;  /* Policy associations
                                       attribute                @01A*/

} EimDomain;

/*-------------------------------------------------------------------*/
/*    EimRegistry                                                    */
/*       List information returned by the following APIs:            */
/*              eimListRegistries                                    */
/*              eimGetRegistryFromAlias                              */
/*-------------------------------------------------------------------*/
typedef struct EimRegistry
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure       */
    enum EimRegistryKind  kind;     /* Kind of registry              */
    EimListData name;               /* Registry name                 */
    EimListData type;               /* Registry type                 */
    EimListData description;        /* Description                   */
    EimListData entryUUID;          /* Entry UUID                    */
    EimListData URI;                /* URI                           */
    EimListData systemRegistryName; /* System registry name          */
    EimSubList  registryAlias;      /* EimRegistryAlias sublist      */
    enum EimStatus  mappingLookup;  /* Mapping lookup attribute   @01A*/
    enum EimStatus  policyAssociations; /* Policy associations
                                       attribute                @01A*/
} EimRegistry;

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

```
/*    EimIdentifier                                                 */
/*       List information returned by the following APIs:           */
/*             eimListIdentifiers                                   */
/*             eimGetAssociatedIdentifiers                          */
/*------------------------------------------------------------------*/
typedef struct EimIdentifier
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure       */
    EimListData uniquename;         /* Unique name                   */
    EimListData description;        /* Description                   */
    EimListData entryUUID;          /* UUID                          */
    EimSubList  names;              /* EimIdentifierName sublist     */
    EimSubList  additionalInfo;     /* EimAddlInfo sublist           */
    enum EimAssociationType type;   /* Association type - only valid
                                  for eimGetAssociatedIdentifiers @02A*/
} EimIdentifier;

/*------------------------------------------------------------------*/
/*    EimAssociation                                                */
/*       List information returned by the following APIs:           */
/*             eimListAssociations                                  */
/*------------------------------------------------------------------*/
typedef struct EimAssociation
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure       */
    enum EimAssociationType associationType; /* Type of association  */
    EimListData registryType;       /* Registry type                 */
    EimListData registryName;       /* Registry name                 */
    EimListData registryUserName;   /* Registry user name            */
} EimAssociation;

/*------------------------------------------------------------------*/
/*    EimRegistryAlias                                              */
/*       List information returned by the following APIs:           */
/*             eimGetRegistryAlias                                  */
/*       Supplemental list information for the following structs:    */
/*             EimRegistry                                          */
/*------------------------------------------------------------------*/
typedef struct EimRegistryAlias
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure       */
    EimListData type;               /* Alias type                    */
    EimListData value;              /* Alias value                   */
} EimRegistryAlias;

/*------------------------------------------------------------------*/
/*    EimRegistryUser                                               */
/*       List information returned by the following APIs:           */
/*             eimListRegistryUsers                                 */
/*------------------------------------------------------------------*/
typedef struct EimRegistryUser
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure       */
    EimListData registryUserName;   /* Name                          */
    EimListData description;        /* Description                   */
    EimSubList  additionalInfo;     /* EimAddlInfo sublist           */
} EimRegistryUser;

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

```
/*  EimTargetIdentity                                                 */
/*     List information returned by the following APIs:               */
/*        eimGetTargetFromSource                                      */
/*        eimGetTargetFromIdentifier                                  */
/*-------------------------------------------------------------------*/
typedef struct EimTargetIdentity
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure       */
    EimListData userName;           /* User name                     */
    enum EimAssociationType type;   /* Association type          @01A*/
} EimTargetIdentity;

/*-------------------------------------------------------------------*/
/*  EimIdentifierName                                                */
/*     Supplemental list information for the following structs:       */
/*        EimIdentifier                                              */
/*-------------------------------------------------------------------*/
typedef struct EimIdentifierName
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure       */
    EimListData name;               /* Name                          */
} EimIdentifierName;
/*-------------------------------------------------------------------*/
/*  EimRegistryName                                                  */
/*     List information returned by the following APIs:               */
/*        eimGetRegistryNameFromAlias                                 */
/*-------------------------------------------------------------------*/
typedef struct EimRegistryName
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure       */
    EimListData name;               /* Name                          */
} EimRegistryName;
/*-------------------------------------------------------------------*/
/*  EimAddlInfo                                                      */
/*-------------------------------------------------------------------*/
/*     Supplemental list information for the following structs:       */
/*        EimRegistryUser                                            */
/*        EimIdentifier                                              */
/*-------------------------------------------------------------------*/
typedef struct EimAddlInfo
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure       */
    EimListData addlInfo;           /* Additional info               */
} EimAddlInfo;

/*-------------------------------------------------------------------*/
/*  EimPolicyFilter                                                  */
/*-------------------------------------------------------------------*/
/*     List information returned by the following APIs:               */
/*        eimListPolicyFilters                                       */
/*-------------------------------------------------------------------*/
typedef struct EimPolicyFilter
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure       */
    enum EimPolicyFilterType type;  /* Type of policy filter.        */
    EimListData sourceRegistry;     /* Source registry name the policy
                                       filter is defined for.        */
```

```
    EimListData filterValue;        /* Filter value.                 */
} EimPolicyFilter;                                          /* @01A*/

/*--------------------------------------------------------------------*/
/*  EimRegistryAssociation                                            */
/*--------------------------------------------------------------------*/
/*     List information returned by the following APIs:               */
/*        eimListRegistryAssociations                                 */
/*--------------------------------------------------------------------*/
typedef struct EimRegistryAssociation
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure        */
    enum EimAssociationType type;   /* Type of association.           */
    EimListData registryName;       /* Registry name the association
                                       is defined to.                 */
    EimListData registryUserName;   /* Registry user name the
                                       association is defined to.     */
    EimListData identifier;         /* Unique name for eim identifier */
    EimListData sourceRegistry;     /* Source registry name the
                                       association is defined for.    */
    EimListData filterValue;        /* Filter value                   */
    enum EimStatus domainPolicyAssocStatus;
                                    /* Policy association status for
                                       the domain:
                                       0 = not enabled
                                       1 = enabled                    */
    enum EimStatus sourceMappingLookupStatus;
                                    /* Mapping lookup status for the
                                       source registry:
                                       0 = not enabled
                                       1 = enabled                    */
    enum EimStatus targetMappingLookupStatus;
                                    /* Mapping lookup status for the
                                       target registry:
                                       0 = not enabled
                                       1 = enabled                    */
    enum EimStatus targetPolicyAssocStatus;
                                    /* Policy association status for
                                       the target registry:
                                       0 = not enabled
                                       1 = enabled                    */
} EimRegistryAssociation;                                   /* @01A*/

/*--------------------------------------------------------------------*/
/*  EimPolicyFilterValue                                              */
/*--------------------------------------------------------------------*/
/*     List information returned by the following APIs:               */
/*        eimFormatPolicyFilter                                       */
/*--------------------------------------------------------------------*/
typedef struct EimPolicyFilterValue
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure        */
    EimListData filterValue;        /* Generated policy filter value  */
} EimPolicyFilterValue;                                     /* @01A*/

/*--------------------------------------------------------------------*/
/*  EimUserIdentity                                                   */
/*--------------------------------------------------------------------*/
/*     Information returned by the following APIs:                    */
/*        eimFormatUserIdentity                                       */
/*--------------------------------------------------------------------*/
typedef struct EimUserIdentity
{
```

```
    unsigned int bytesReturned;     /* Number of bytes actually
                                       by the API.                  */
    unsigned int bytesAvailable;    /* Number of bytes of available
                                       data that could be returned by
                                       the API.                     */
    EimListData userIdentity;       /* User identity                */
} EimUserIdentity;                                          /* @01A*/

/*--------------------------------------------------------------------*/
/*  EimAccess                                                         */
/*--------------------------------------------------------------------*/
/*     List information returned by the following APIs:               */
/*        eimListAccess                                               */
/*--------------------------------------------------------------------*/
typedef struct EimAccess
{
    unsigned int nextEntry;         /* Displacement to next entry. This
                                       byte offset is relative to the
                                       start of this structure       */
    EimListData user;               /* User with access. This data will
                                       be in the format of the dn for
                                       for access id                 */
} EimAccess;

/*--------------------------------------------------------------------*/
/*  EimUserAccess                                                     */
/*--------------------------------------------------------------------*/
/*     List information returned by the following APIs:               */
/*        eimListUserAccess                                           */
/*--------------------------------------------------------------------*/
typedef struct EimUserAccess
{
    unsigned int nextEntry;         /* Displacement to next entry. This
                                       byte offset is relative to the
                                       start of this structure       */
    enum EimAccessIndicator eimAdmin;
    enum EimAccessIndicator eimRegAdmin;
    enum EimAccessIndicator eimIdenAdmin;
    enum EimAccessIndicator eimMappingLookup;
    EimSubList  registries;         /* EimRegistryName sublist       */
} EimUserAccess;

/*--------------------------------------------------------------------*/
/*                                                                    */
/*    Domain                                                          */
/*                                                                    */
/*--------------------------------------------------------------------*/

int eimCreateDomain
   (
    char           * ldapURL,       /* Input:  ldap URL that indicates
                                       host, port, parent dn         */
    EimConnectInfo   connectInfo,   /* Input: Connection information  */
    char           * description,   /* Input: Domain description      */
    EimRC          * eimrc          /* Input/Output: return code      */
    );

int eimDeleteDomain
   (
    char           * ldapURL,       /* Input:  ldap URL that indicates
                                       host, port, parent dn         */
    EimConnectInfo   connectInfo,   /* Input: Connection information  */
    EimRC          * eimrc          /* Input/Output: return code      */
    );

int eimChangeDomain
   (
```

```
    char           * ldapURL,       /* Input:  ldap URL that indicates
                                       host, port, parent dn          */
    EimConnectInfo   connectInfo,   /* Input: Connection information  */
    enum EimDomainAttr attrName,    /* Input: Attribute to change     */
    char           * attrValue,     /* Input: New attribute value     */
    enum EimChangeType changeType,  /* Input: Type of change          */
    EimRC          * eimrc          /* Input/Output: return code      */
    );


int eimListDomains
   (
    char           * ldapURL,       /* Input:  ldap URL that indicates
                                       host, port, parent dn          */
    EimConnectInfo   connectInfo,   /* Input: Connection information  */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                       listData                       */
    EimList        * listData,      /* Output:  In EimList the field
                                       firstEntry will get to the
                                       first EimDomain element.       */
    EimRC          * eimrc          /* Input/Output: return code      */
    );
/*--------------------------------------------------------------------*/
/*                                                                    */
/*    Configuration                                                   */
/*                                                                    */
/*--------------------------------------------------------------------*/
int eimSetConfiguration
   (
    int              enable,        /* Input:  indicate if enabled to
                                       participate in EIM domain
                                       0 = not enabled
                                       1 = enabled                    */
    char           * ldapURL,       /* Input: LDAP URL configuration
                                       information: host, port and
                                       domain dn                      */
    char           * localRegistry, /* Input: Local registry name     */
    char           * kerberosRegistry, /* Input:  Kerberos registry   */
    int              ccsid,         /* CCSID of the input data        */
    EimRC          * eimrc          /* Input/Output: return code      */
    );

int eimSetConfigurationExt
   (
    EimConfigInfo  * configInfo,    /* Input:  configuration info.    */
    EimRC          * eimrc          /* Input/Output: return code      */
    );                                                    /* @02A*/

int eimRetrieveConfiguration
   (
    unsigned int     lengthOfEimConfig, /* Input: size provided for
                                       configData                     */
    EimConfig      * configData,    /* Output: Configuration data
                                       returned.                      */
#ifdef __MVS__                                             /*@L4A*/
    char           * profile,       /* Input: Name of profile that
                                       contains z/OS config info. @L4A*/
    char           * userIdentity,  /* Input: User Id with an LDAPBIND
                                       class profile.             @L4A*/
#endif                                                     /*@L4A*/
    int              ccsid,         /* CCSID the data will be returned
                                       in                             */
    EimRC          * eimrc          /* Input/Output: return code      */
    );
/*--------------------------------------------------------------------*/
/*                                                                    */
/*    Handles                                                         */
```

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

int eimCreateHandle
   (
    EimHandle      * eim,           /* Output: eimHandle             */
    char           * ldapURL,       /* Input:  ldap URL that indicates
                                       host, port, parent dn         */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimDestroyHandle
   (
    EimHandle      * eim,           /* Input: eimHandle              */
    EimRC          * eimrc          /* Input/Output: return code     */
   );

int eimGetAttribute
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    enum EimHandleAttr attrName,    /* Input: name of attribute to get
                                                                     */
    unsigned int lengthOfEimAttribute, /* Input: size provided for
                                       EimAttribute                  */
    EimAttribute  * attribute,      /* Output: Attribute data
                                       returned.                     */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimSetAttribute
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    enum EimHandleAttr attrName,    /* Input: name of attribute to set
                                                                     */
    void           * attrValue,     /* Input: Pointer to buffer to
                                       the new attribute value       */
    EimRC          * eimrc          /* Input/Output: return code     */
    );


int eimGetVersion
   (
    EimHostInfo     * hostInfo,     /* Input: Host information       */
    enum EimVersion * version,      /* Output:  version number       */
    EimRC           * eimrc         /* Input/Output: return code     */
    );                              /*                            @08A*/


/*------------------------------------------------------------------*/
/*                                                                  */
/*   Connect                                                        */
/*                                                                  */
/*------------------------------------------------------------------*/
int eimConnect
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    EimConnectInfo   connectInfo,   /* Input: Connection information  */
    EimRC          * eimrc          /* Input/Output: return code     */
    );
int eimConnectToMaster
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    EimConnectInfo   connectInfo,   /* Input: Connection information  */
    EimRC          * eimrc          /* Input/Output: return code     */
    );
```

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

int eimAddSystemRegistry
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    char           * registryName,  /* Input: Registry name          */
    char           * registryType,  /* Input: Registry type          */
    char           * description,   /* Input: Description            */
    char           * URI,           /* Input: URI                    */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimAddApplicationRegistry
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    char           * registryName,  /* Input: Registry name          */
    char           * registryType,  /* Input: Registry type          */
    char           * description,   /* Input: Description            */
    char           * systemRegistryName, /* Input: Associated system
                                       registry                      */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimRemoveRegistry
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    char           * registryName,  /* Input: Registry name          */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimChangeRegistry
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    char           * registryName,  /* Input: Registry name          */
    enum EimRegistryAttr attrName,  /* Input: name of attribute to
                                       change.                       */
    char           * attrValue,     /* Input: new value for attribute */
    enum EimChangeType changeType,  /* Input: Type of change to make  */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimListRegistries
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    char           * registryName,  /* Input: Registry name          */
    char           * registryType,  /* Input: Registry type          */
    enum EimRegistryKind registryKind,/* Input: Registry kind        */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                       listData                      */
    EimList        * listData,      /* Output:  In EimList the field
                                       firstEntry will get to the
                                       first EimRegistry element     */
    EimRC          * eimrc          /* Input/Output: return code     */
    );


/*-------------------------------------------------------------------*/
/*                                                                   */
/*    Identifier                                                     */
/*                                                                   */
/*-------------------------------------------------------------------*/

int eimAddIdentifier
```

```
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    char           * name,          /* Input: Requested name for
                                       Identifier                    */
    enum EimIdAction nameInUseAction, /* Input: Action to take if the
                                      requested name is already in use */
    unsigned int * sizeOfUniqueName, /* Input/Output: size of
                                       uniqueName field              */
    char           * uniqueName,    /* Output: Unique name           */
    char           * description,   /* Input: Description            */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimRemoveIdentifier
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    EimIdentifierInfo * idName,     /* Input: Identifier info        */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimChangeIdentifier
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    EimIdentifierInfo * idName,     /* Input: Identifier info        */
    enum EimIdentifierAttr attrName, /* Input: name of attribute to
                                       change.                       */
    char           * attrValue,     /* Input: new value for attribute */
    enum EimChangeType changeType,  /* Input: Type of change to make  */
    EimRC          * eimrc          /* Input/Output: return code     */

    );
int eimListIdentifiers
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    EimIdentifierInfo * idName,     /* Input: Identifier info        */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                       listData                      */
    EimList        * listData,      /* Output:  In EimList the field
                                       firstEntry will get to the
                                       first EimIdentifier element   */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimGetAssociatedIdentifiers
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    enum EimAssociationType associationType, /* Input: Type of
                                       association                   */
    char           * registryName,  /* Input: Registry name          */
    char           * registryUserName, /* Input: Registry user name   */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                       listData                      */
    EimList        * listData,      /* Output:  In EimList the field
                                       firstEntry will get to the
                                       first EimIdentifier element   */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

/*--------------------------------------------------------------------*/
/*                                                                    */
/*    Association                                                     */
/*                                                                    */
/*--------------------------------------------------------------------*/

int eimAddAssociation
   (
    EimHandle      * eim,           /* Input: Eim handle             */
```

```
    enum EimAssociationType associationType, /* Input: Type of
                                        association                   */
    EimIdentifierInfo * idName,     /* Input: Identifier info        */
    char           * registryName,  /* Input: Registry name          */
    char           * registryUserName, /* Input: Registry user name  */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimRemoveAssociation
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    enum EimAssociationType associationType, /* Input: Type of
                                        association                   */
    EimIdentifierInfo * idName,     /* Input: Identifier info        */
    char           * registryName,  /* Input: Registry name          */
    char           * registryUserName, /* Input: Registry user name  */
    EimRC          * eimrc          /* Input/Output: return code     */
    );


int eimListAssociations
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    enum EimAssociationType associationType, /* Input: Type of
                                        association                   */
    EimIdentifierInfo * idName,     /* Input: Identifier info        */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                        listData                      */
    EimList        * listData,      /* Output:  In EimList the field
                                        firstEntry will get to the
                                        first EimAssociation element  */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimListRegistryAssociations
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    enum EimAssociationType associationType,  /* Input: Type of policy
                                        association                   */
    char           * registryName,  /* Input: Registry name          */
    char           * registryUserName, /* Input: Registry user name  */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                        listData                      */
    EimList        * listData,      /* Output:  In EimList the field
                                        firstEntry will get to the
                                        first EimRegistryAssociation
                                        element                       */
    EimRC          * eimrc          /* Input/Output: return code     */
   );                                                      /* @01A*/


/*-------------------------------------------------------------------*/
/*                                                                   */
/*    Mappings                                                       */
/*                                                                   */
/*-------------------------------------------------------------------*/
int eimGetTargetFromSource
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    char           * sourceRegistryName,/* Input: Source registry name
                                                                     */
    char           * sourceRegistryUserName,/* Input: Source registry
                                        user name                     */
    char           * targetRegistryName, /* Input: Target registry name
                                                                     */
    char           * additionalInformation, /* Input: Additional info */
    unsigned int     lengthOfListData, /* Input:  size provided for
```

```
                                      listData                      */
    EimList        * listData,      /* Output:  In EimList the field
                                      firstEntry will get to the
                                      first EimTargetIdentity element*/
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimGetTargetFromIdentifier
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    EimIdentifierInfo * idName,     /* Input: Identifier info        */
    char           * targetRegistryName, /* Input: Target registry name
                                                                     */
    char           * additionalInformation, /* Input: Additional info */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                      listData                      */
    EimList        * listData,      /* Output:  In EimList the field
                                      firstEntry will get to the
                                      first EimTargetIdentity element*/
    EimRC          * eimrc          /* Input/Output: return code     */
    );


/*--------------------------------------------------------------------*/
/*                                                                    */
/*    Registry User                                                   */
/*                                                                    */
/*--------------------------------------------------------------------*/
int eimChangeRegistryUser
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    char           * registryName,  /* Input: Registry name          */
    char           * registryUserName, /* Input: Registry user name   */
    enum EimRegistryUserAttr attrName, /* Input: name of attribute to
                                       change.                      */
    char           * attrValue,     /* Input: new value for attribute */
    enum EimChangeType changeType,  /* Input: Type of change to make  */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimListRegistryUsers
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    char           * registryName,  /* Input: Registry name          */
    char           * registryUserName, /* Input: Registry user name   */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                      listData                      */
    EimList        * listData,      /* Output:  In EimList the field
                                      firstEntry will get to the
                                      first EimRegistryUser element  */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

/*--------------------------------------------------------------------*/
/*                                                                    */
/*    Registry Alias                                                  */
/*                                                                    */
/*--------------------------------------------------------------------*/
int eimChangeRegistryAlias
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    char           * registryName,  /* Input: Registry name          */
    char           * aliasType,     /* Input: Registry alias type    */
    char           * aliasValue,    /* Input: Registry alias value   */
    enum EimChangeType changeType,  /* Input: Type of change to make  */
    EimRC          * eimrc          /* Input/Output: return code     */
    );
```

```
int eimListRegistryAliases
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    char           * registryName,  /* Input: Registry name          */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                      listData                      */
    EimList        * listData,      /* Output:  In EimList the field
                                      firstEntry will get to the
                                      first EimRegistryAlias element */
    EimRC          * eimrc          /* Input/Output: return code     */
    );

int eimGetRegistryNameFromAlias
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    char           * aliasType,     /* Input: Registry alias type    */
    char           * aliasValue,    /* Input: Registry alias value   */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                      listData                      */
    EimList        * listData,      /* Output:  In EimList the field
                                      firstEntry will get to the
                                      first EimRegistryName element  */
    EimRC          * eimrc          /* Input/Output: return code     */
    );



/*--------------------------------------------------------------------*/
/*                                                                    */
/*    Policies                                                        */
/*                                                                    */
/*--------------------------------------------------------------------*/
int eimFormatPolicyFilter
   (
    EimUserIdentityInfo * userIdentityInfo, /* Input: User identity
                                      information to format         */
    EimPolicyFilterSubsetInfo * subsetInfo, /* Input: Subset info    */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                      listData                      */
    EimList        * listData,      /* Output:  In EimList the field
                                      firstEntry will get to the
                                      first EimPolicyFilterValue
                                      element                       */
    EimRC          * eimrc          /* Input/Output: return code     */
   );                                                    /* @01A*/

int eimAddPolicyFilter
   (
    EimHandle           * eim,      /* Input: Eim handle             */
    EimPolicyFilterInfo * filterInfo, /* Input: Policy filter info   */
    EimRC               * eimrc     /* Input/Output: return code     */
   );                                                    /* @01A*/

int eimRemovePolicyFilter
   (
    EimHandle           * eim,      /* Input: Eim handle             */
    EimPolicyFilterInfo * filterInfo, /* Input: Policy filter info   */
    EimRC               * eimrc     /* Input/Output: return code     */
   );                                                    /* @01A*/

int eimListPolicyFilters
   (
    EimHandle      * eim,           /* Input: Eim handle             */
    enum EimPolicyFilterType filterType,  /* Input: Type of policy
                                      filter                        */
```

```
    char           * registryName,  /* Input: Registry name           */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                       listData                       */
    EimList        * listData,      /* Output:  In EimList the field
                                       firstEntry will get to the
                                       first EimPolicyFilter element  */
    EimRC          * eimrc          /* Input/Output: return code      */
   );                                                    /* @01A*/

int eimAddPolicyAssociation
   (
    EimHandle           * eim,      /* Input: Eim handle              */
    EimPolicyAssociationInfo * policyAssoc, /* Input: Policy
                                       association info               */
    EimRC               * eimrc     /* Input/Output: return code      */
   );                                                    /* @01A*/

int eimRemovePolicyAssociation
   (
    EimHandle           * eim,      /* Input: Eim handle              */
    EimPolicyAssociationInfo * policyAssoc, /* Input: Policy
                                       association info               */
    EimRC               * eimrc     /* Input/Output: return code      */
   );                                                    /* @01A*/



/*--------------------------------------------------------------------*/
/*                                                                    */
/*    User Identities                                                 */
/*                                                                    */
/*--------------------------------------------------------------------*/
int eimFormatUserIdentity
   (
    enum EimUserIdentityFormatType formatType,  /* Input: Type of format
                                       to return                      */
    EimUserIdentityInfo * userIdentityInfo, /* Input: User identity
                                       information to format          */
    unsigned int      lengthOfUserIdentity, /* Input:  size provided for
                                       userIdentity                   */
    EimUserIdentity * userIdentity, /* Output:  formatted user
                                       identity                       */
    EimRC           * eimrc         /* Input/Output: return code      */
   );                                                    /* @01A*/



/*--------------------------------------------------------------------*/
/*                                                                    */
/*    Access                                                          */
/*                                                                    */
/*--------------------------------------------------------------------*/
int eimAddAccess
   (
    EimHandle      * eim,           /* Input: Eim handle              */
    EimAccessUser  * accessUser,    /* Input: User for access         */
    enum EimAccessType accessType,  /* Input: Type of access          */
    char           * registryName,  /* Input: Registry name           */
    EimRC          * eimrc          /* Input/Output: return code      */
    );

int eimRemoveAccess
   (
    EimHandle      * eim,           /* Input: Eim handle              */
    EimAccessUser  * accessUser,    /* Input: User for access         */
    enum EimAccessType accessType,  /* Input: Type of access          */
    char           * registryName,  /* Input: Registry name           */
```

```
    EimRC          * eimrc          /* Input/Output: return code      */
    );
int eimListAccess
   (
    EimHandle      * eim,           /* Input: Eim handle              */
    enum EimAccessType accessType,  /* Input: Type of access          */
    char           * registryName,  /* Input: Registry name           */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                      listData                       */
    EimList        * listData,      /* Output:  In EimList the field
                                      firstEntry will get to the
                                      first EimAccess element        */
    EimRC          * eimrc          /* Input/Output: return code      */
    );
int eimListUserAccess
   (
    EimHandle      * eim,           /* Input: Eim handle              */
    EimAccessUser  * accessUser,    /* Input: User for access         */
    unsigned int     lengthOfListData, /* Input:  size provided for
                                      listData                       */
    EimList        * listData,      /* Output:  In EimList the field
                                      firstEntry will get to the
                                      first EimUserAccess element    */
    EimRC          * eimrc          /* Input/Output: return code      */
    );
int eimQueryAccess
   (
    EimHandle      * eim,           /* Input: Eim handle              */
    EimAccessUser  * accessUser,    /* Input: User for access         */
    enum EimAccessType accessType,  /* Input: Type of access          */
    char           * registryName,  /* Input: Registry name           */
    unsigned int * accessIndicator, /* Output:  Indicates
                                      whether access found           */
    EimRC          * eimrc          /* Input/Output: return code      */
    );


/*--------------------------------------------------------------------*/
/*                                                                    */
/*    Error Message                                                   */
/*                                                                    */
/*--------------------------------------------------------------------*/
char * eimErr2String
   (
    EimRC          * eimrc          /* Input: return code             */
    );


#pragma enum(pop)

#ifdef __cplusplus
}
#endif

#if (__OS400_TGTVRM__>=510)
#pragma datamodel(pop)
#endif

#endif /* EIM_h */
```

Example for creating LDAP suffix and user objects

The following is a sample **ldapadd** command and ldif file to create suffix objects and an LDAP user object. Here are the suffix.ldif file contents:

```
# ------------------------------------------------------------------
# Create the country us object
# ------------------------------------------------------------------
dn: c=us
objectclass: top
objectclass: country
c: us

# ------------------------------------------------------------------
# Create the ibm organization under c=us
# ------------------------------------------------------------------
dn: o=ibm,c=us
objectclass: top
objectclass: organization
o: ibm

# ------------------------------------------------------------------
# Create the dept20 organizational unit object under
#  o=ibm,c=us
# ------------------------------------------------------------------
dn: ou=dept20,o=ibm,c=us
objectclass: top
objectclass: organizationalunit
ou: dept20

# ------------------------------------------------------------------
# Create the eim administrator user object with a password
# of "secret" under ou=dept20,o=ibm,c=us
# ------------------------------------------------------------------
dn: cn=eim administrator,ou=dept20,o=ibm,c=us
objectclass: top
objectclass: person
sn: eim administrator
cn: eim administrator
userpassword: secret

# End of file suffix.ldif
```

Then, to add these entries to LDAP, issue the following **ldapadd** command from the z/OS UNIX shell:

```
ldapadd
-h ldap://some.ldap.host
-D cn=ldap administrator
-w secret
-f suffix.ldif
```

Part 2. Working with remote services

Chapter 13. The z/OS Identity Cache

Today's transaction environments are typically multi-platform heterogeneous configurations with disparate security domains. Each security domain can have its own security service provider and authentication methods. While a user might be authenticated in one security domain, the authenticated identity information may be lost when transactions initiated in one security domain are executed in another.

For example, a user may authenticate his or her identity to a security product running in a distributed environment, and then initiate a transaction that executes on a z/OS system under the authority of a shared identity. Since the user's original authenticated identity is not known to z/OS, SMF audit records contain information only for the shared identity. Because the original authenticated identity is not maintained across the security domains, individual accountability is lost.

The z/OS Identity Cache enables an application to maintain identity information across security domains, so that individual accountability is not lost. Distributed applications can use the z/OS Identity Cache to enable end-to-end auditing that tracks the identity initially used for authentication as well as the identity on the current platform.

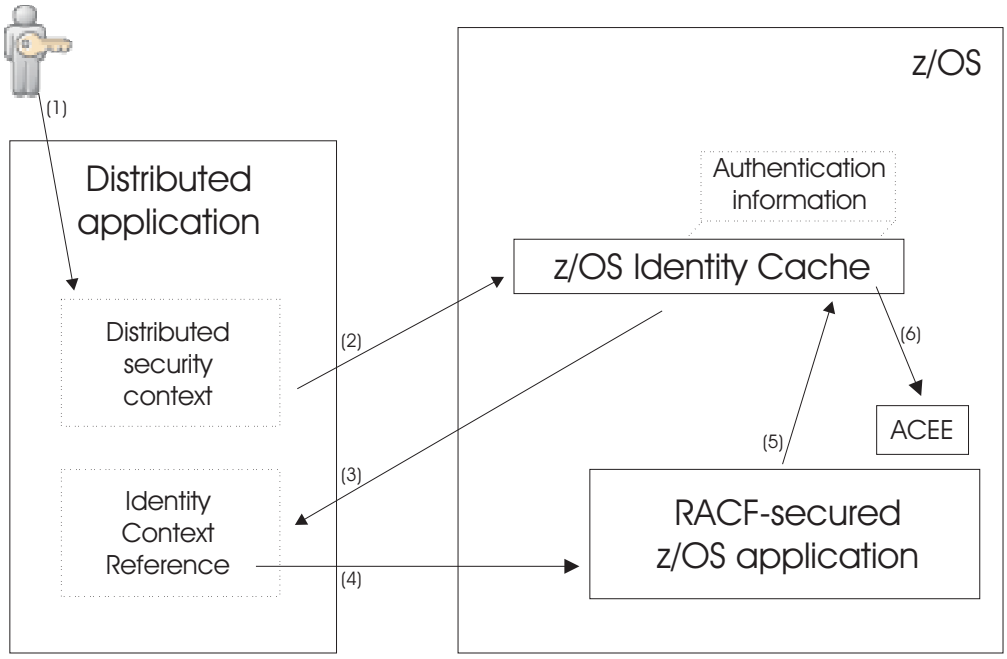
The primary interface to the z/OS Identity Cache is the ICTX Java API which enables applications to store identity context information in the Identity Cache on the local or on a remote z/OS system. Remote access to the Identity Cache from a z/OS or non-z/OS system is provided by LDAP extended operations support which is leveraged by the ICTX Java API. Cache management is provided by the R_cacheserv SAF callable service.

The ICTX Java API method for storing identity context information in the Identity Cache returns a reference which can subsequently be used to retrieve that information. For example, a distributed application could pass the reference to a RACF-secured z/OS application which could then retrieve the original user authentication information from the Identity Cache. In particular, the RACROUTE REQUEST = VERIFY macro and the initACEE SAF callable service both accept an identity context reference as a user ID and password and will create an access control environment element (ACEE) that includes the identity context information from the Identity Cache. This information will in turn be included in SMF audit records.

How the z/OS Identity Cache works

An application can store user authentication information in the z/OS Identity Cache, and will get an 8-byte identity context reference value which can subsequently be used to retrieve that information. The reference can be passed to an application running on z/OS, so that when it uses the RACROUTE REQUEST=VERIFY macro or the initACEE service to create a z/OS security context, the ACEE created will be extended to include the original authentication information. Because the original authentication information is now included in the ACEE, SMF audit records for the user will also contain this extended authentication information.

The following figure illustrates how a distributed application can use the z/OS Identity Cache to make the original user authentication information available to a remote z/OS application and system.



In this figure, the user (1) authenticates to a distributed application. The distributed application uses the ICTX Java API to (2) store the authentication information in a the z/OS Identity Cache on a remote z/OS system. The ICTX Java API (3) returns the identity context reference to the calling application, which in turn (4) passes this reference to a RACF-secured z/OS application. The z/OS application uses the identity context reference in place of a user ID and password, when it (5) calls the RACROUTE REQUEST=VERIFY macro or the initACEE SAF callable service to build the security context for the user. RACF will recognize that the supplied user ID and password is an identity context reference, and so will (6) retrieve the user authentication information from the z/OS Identity Cache and will build an ACEE that contains the extended authentication information for the user. This extended authentication information from the Identity Cache will be included in subsequent SMF audit records for transactions initiated by the user.

The application accessing a z/OS Identity Cache may be running locally or remotely. Remote applications running on either a z/OS or non-z/OS system can access an Identity Cache on a remote z/OS system using a z/OS IBM TDS server configured with ICTX extended operations. The application will provide the ICTX Java API with a URL identifying the z/OS IBM TDS server as well as the ID and password for a RACF user ID on the remote system. The z/OS IBM TDS server will perform a simple bind operation using the supplied RACF ID and password for authentication to the remote z/OS system. For z/OS applications accessing the local Identity Cache, the z/OS IBM TDS server is not needed.

The Identity Cache can be configured to ensure that the stored information on the authenticated user also contains a mapping to a local z/OS user ID. The Identity Cache can be configured to accept a mapping supplied by the application, or it can be configured to access an EIM domain to perform a lookup operation to find the mapping. The local user ID, whether provided by the application or identified using an EIM lookup operation, will be stored in the Identity Cache along with the initial authentication information. If a reference to an identity context in this Identity Cache is passed to the RACROUTE REQUEST=VERIFY macro or the initACEE SAF callable service, the resulting ACEE, and subsequent SMF audit records, will also include the additional mapped local user ID information.

Configuring your environment to use the z/OS Identity Cache

Some configuration is required for z/OS systems that host, and applications that use, an Identity Cache. The authenticating application needs to know where the Identity Cache is located so that it can store information about the end user and get back an identity context reference. The authenticating application then passes the identity context reference to a server which also needs to know where the Identity Cache resides so that it can retrieve the stored information. An application will need to know if it is accessing the Identity Cache on a local node, or if it accessing the Identity Cache on a remote node. If accessing the Identity Cache on a remote node, the application will need to know the host name of the z/OS IBM TDS server that is providing the connection to the remote system.

If you want the Identity Cache to contain user ID mappings from the authenticated user ID to a local z/OS user ID, you need to configure the Identity Cache to specify that ID mapping is required. You can also configure the Identity Cache to specify how it should obtain these mappings – from the application storing the authenticated user ID information in the Identity Cache, from an EIM lookup operation, or using a combination of these approaches. If the Identity Cache is configured to do an EIM lookup, an EIM domain will also need to be set up.

If you are using the Identity Cache on more than one system in a z/OS sysplex, there are some additional configuration considerations. See “Configuring z/OS sysplex for the Identity Cache” on page 430 for more information.

Configuring Java applications to use the z/OS Identity Cache

The main application interface to the z/OS Identity Cache is the ICTX Java API described in more detail in Chapter 14, “ICTX Java API,” on page 431. Applications can use the API to access the Identity Cache on the local z/OS system or on a remote z/OS system.

To use the ICTX Java API to access a z/OS Identity Cache on either the local or a remote z/OS system, all applications must:

- have access to the ICTX Java classes defined in the ictx.jar file, which is located in the `/usr/lpp/eim/lib` HFS directory. To get access to the ICTX Java classes, include the ictx.jar file in the CLASSPATH of the Java application.
- have a RACF user ID on the z/OS system where the Identity Cache is located, and the permission necessary to access the Identity Cache. The ICTX Java API uses the R_cacheserv callable service to perform read and write operations to the Identity Cache. Use of the R_cacheserv callable service is authorized by the IRR.RCACHESERV.ICTX resource in the FACILITY class, so the RACF user ID associated with the request to access the Identity Cache must have the correct access permission to the IRR.RCACHESERV.ICTX resource in the FACILITY class.
  - If the application is a z/OS application accessing the Identity Cache on the local system, then the RACF user ID associated with the request is the one under which the application is running.
  - If the application is running on a z/OS or non-z/OS system and accessing the Identity Cache on a remote z/OS system, it will be accessing the remote system's Identity Cache through an IBM TDS server. In this case, the RACF user ID associated with the request is the RACF user ID used to authenticate with the remote z/OS system through an LDAP bind operation.

The level of access permission required to access the IRR.RCACHESERV.ICTX resource in the FACILITY class depends on the specific type of operation the application needs to perform.

– If the application is going to store user information in the Identity Cache, then the RACF user ID needs UPDATE access to the IRR.RCACHESERV.ICTX resource in the FACILITY class.
– If the application is going to retrieve information from the Identity Cache (using either the ICTX Java API, the RACROUTE REQUEST=VERIFY macro, or the initACEE SAF callable service), then the RACF user ID needs READ access to the IRR.RCACHESERV.ICTX resource in the FACILITY class.

The FACILITY class must be active and enabled for RACLIST processing. Access updates for a user will not take effect until the user logs on again.

In addition to the preceding configuration requirements common to all Java applications, additional configuration requirements vary depending on whether the application is accessing the Identity Cache on the local or on a remote z/OS system.

- If the application will be accessing the Identity Cache on the local z/OS system, it must also have access to the EIM and ICTX API routines. To get access to these routines, include the HFS directory `/usr/lpp/eim/lib` in the LIBPATH of the z/OS Java application.
- If the application will be accessing the Identity Cache on a remote z/OS system, it must also specify:
  – the host name and, optionally, the port number for the z/OS IBM TDS server that is providing the connection to the z/OS Identity Cache on the remote system. This host name may begin with ldap:// or ldaps://. If ldaps:// is used, then it is assumed the IBM TDS server as been set up for SSL or TLS communication.
    - If the application is running on a z/OS system, the host name for the z/OS IBM TDS server can be set as an in-storage Identity Cache default value (as described in “Configuring Identity Cache connection defaults” on page 429).
    - If the application is running on a non-z/OS system, the host name for the z/OS IBM TDS server could be obtained from a properties file specific to the application.
  – the bind credentials for connecting to the remote system. The bind credentials correspond to a RACF user ID and password on the remote system. If the application is running on a z/OS system, the bind credentials can be set as in-storage Identity Cache default values (as described in “Configuring Identity Cache connection defaults” on page 429).

  The z/OS IBM TDS server needs to have been configured with ICTX extended operations and started. To configure the z/OS IBM TDS server with ICTX extended operations, modify the ds.conf file with a section that defines the ICTX extended operations support. For more information, refer to “Configuring the IBM Tivoli Directory Server for remote services support” on page 432.

Configuring the z/OS Identity Cache

You can configure the Identity Cache to:

- determine if, and how, mappings to local z/OS user IDs will be included in the Identity Cache. These user ID mappings can be provided by applications, or determined by an EIM lookup operation. The Identity Cache can be configured so that valid ID mappings are:

  - not stored, even if provided by applications
  - optionally stored if provided by an application or identified using EIM
  - required, so that the operation to store information in the Identity Cache will fail if a valid ID mapping is not provided by an application or identified using EIM.

  An expiration time for stored mappings can also be configured.
- specify, for applications running on z/OS, defaults for connecting to the Identity Cache.

Configuring user ID mapping

You can configure the Identity Cache to specify how mappings to local z/OS user IDs should be handled. These configuration options can be specified in the IRR.ICTX.DEFAULTS.*sysid* or IRR.ICTX.DEFAULTS profile in the LDAPBIND class.

- The USEMAP configuration option specifies whether to accept or ignore user ID mappings provided by applications. If you specify USEMAP(YES), and an application provides a valid mapping to a local z/OS user ID, it will be stored in the Identity Cache. If you specify USEMAP(NO), application-provided ID mappings will be ignored.
- The DOMAP configuration option specifies whether or not EIM services should be used to map a user ID stored by the application in the Identity Cache to a local z/OS user ID. If you specify DOMAP(YES), an EIM lookup operation will be used to find the ID mapping. If a mapping is found, it will be stored in the Identity Cache. If you specify DOMAP(NO), the Identity Cache will not use EIM to find a mapping.

  If DOMAP(YES) is used, the Identity Cache needs to be fully configured to access the EIM domain. See “Configuring and setting up EIM” on page 429 for more information.
- The MAPREQUIRED configuration option specifies whether or not a mapping to a local z/OS user ID is required. If MAPREQUIRED(YES) is specified and no valid mapping is provided by the application or found using EIM, the application request to store information in the Identity Cache will fail. If MAPREQUIRED(NO) is specified, then valid ID mappings will, if provided by the application or found using EIM, be stored in the Identity Cache, but will not be required.

The following are the default user ID mapping settings:

- USEMAP(YES) — user ID mappings provided by applications are accepted
- DOMAP(NO) — the Identity Cache does not perform mapping itself
- MAPREQUIRED(NO) — there is no requirement for a user ID mapping to be stored with user information

The following table summarizes the combined effect of the various configuration option settings.

Table 48. user ID mapping configuration settings

USEMAP	DOMAP	MAPREQUIRED	Effect
NO	NO	NO	The Identity Cache will not use an application provided mapping to a z/OS user ID and it will not use EIM to find a mapping. Even if an application mapping is provided, no mapping is stored.

Table 48. user ID mapping configuration settings (continued)

USEMAP	DOMAP	MAPREQUIRED	Effect
NO	NO	YES	The Identity Cache will not use an application provided mapping to a z/OS user ID and it will not use EIM to find a mapping. A mapping is required, so the Identity Cache will return an error.
NO	YES	NO	The Identity Cache will not use an application provided mapping to a z/OS user ID. It will use EIM to find a mapping. If a mapping is found, it will be stored. If no mapping is found, the Identity Cache will not return an error.
NO	YES	YES	The Identity Cache will not use an application provided mapping to a z/OS user ID. It will use EIM to find a mapping. If a mapping is found, it will be stored. If no mapping is found, the Identity Cache will return an error.
YES	NO	NO	If an application provides a mapping to a z/OS user ID, it will be stored. The Identity Cache will not use EIM to find a mapping. If no mapping is provided, the Identity Cache will not return an error.
YES	NO	YES	If an application provides a mapping to a z/OS user ID, it will be stored. The Identity Cache will not use EIM to find a mapping. If no mapping is provided, the Identity Cache will return an error.
YES	YES	NO	If an application provides a mapping to a z/OS user ID, the Identity Cache will store it. If not, the Identity Cache will use EIM to find a mapping to store. If no mapping is provided or found, the Identity Cache will not return an error.
YES	YES	YES	If an application provides a mapping to a z/OS user ID, the Identity Cache will store it. If not, the Identity Cache will use EIM to find a mapping to store. If no mapping is provided or found, the Identity Cache will return an error.

Since user ID mappings can change over time, a user ID mapping stored in the Identity Cache will expire. By default, a user ID mapping will be stored in the Identity Cache for 1 hour, but you can set this to a shorter interval using the MAPPINGTIMEOUT configuration option. The MAPPINGTIMEOUT interval is specified in seconds, and valid values range from 1 to 3600.

A copy of the current configuration settings is kept in-storage to allow fast access to the values. The configuration settings can be defined in an IRR.ICTX.DEFAULTS.*sysid* profile in the LDAPBIND class. For example:

```
RDEFINE LDAPBIND IRR.ICTX.DEFAULTS.SYSA ICTX(USEMAP(NO) DOMAP(YES)
MAPREQUIRED(YES) MAPPINGTIMEOUT(1800)) EIM(LOCALREGISTRY(MyZOSRegistry))
```

This example defines the settings for the system with a system ID of SYSA, and establishes the following rules:

- Because USEMAP(NO) is specified, any mappings provided by the application are ignored.

- Because DOMAP(YES) is specified, the Identity Cache itself will extract mappings from EIM. Because LOCALREGISTRY(MyZOSRegistry) is specified, the Identity Cache will use the EIM registry name of MyZOSRegistry when looking for mappings in the configured EIM domain.
- Because MAPREQUIRED(YES) is specified, any requests to store user identity information are rejected if a mapping can't be found in EIM.
- Because MAPPINGTIMEOUT(1800) is specified, the mappings to RACF user IDs that are stored in the cache can be reused for 30 minutes instead of the default 1 hour.

These same values could be stored in a profile with the name IRR.ICTX.DEFAULTS. However, be careful using this because applications on other systems that are sharing the RACF database may also default to using this same profile.

The final step to making the settings take effect is that the LDAPBIND class must be active and RACLISTed. The entire set of field values must be specified or defaulted to in the profile before the in-storage copy of the fields is updated.

Configuring and setting up EIM: When the Identity Cache is configured to do an EIM lookup; an EIM domain needs to be set up. This domain is a directory on an IBM TDS server which contains EIM data for an enterprise, and is a collection of all EIM identifiers, associations, and user registries that are defined in that location. EIM clients participate in the domain by using the domain data for EIM lookup operations. In order for Identity Cache to do a lookup, the EIM domain must contain the required data for mapping a user identity defined remotely to a user identity defined locally. The IBM TDS server that hosts your EIM domain can run on a z/OS or non-z/OS system. See Part 1, "EIM concepts and use," on page 1 for more information on how to set up your EIM domain.

Once you have set up your EIM domain, you will also need to use RACF commands to give the Identity Cache access to the EIM domain by defining the domain name, the distinguished name, and password which will be used for IBM TDS binding. See "Setting up default domain LDAP URL and binding information" on page 68 for more information about how to do this.

If you choose to create a profile for IBM TDS binding information, you will need to update the RACF user ID with this information. The user ID you will need to update depends on whether the request is local or remote. If the request is remote, you will need to update the user ID used in the LDAP bind operation. If the request is local, you will need to update the user ID under which the application is running. For example, to update the RACF user ID IBMUSER, you would use the following command:

```
ALU IBMUSER EIM(LDAPPROF("+ racfProfile +"))
```

Configuring Identity Cache connection defaults

Applications running on z/OS and using the Identity Cache may choose to use the configured Identity Cache for that system. That cache can reside on another z/OS box or on the local z/OS system. To access a cache running on a remote system, the command that would need to be issued is:

```
RDEFINE LDAPBIND IRR.ICTX.DEFAULTS.sysid PROXY(LDAPHOST(hostname)
BINDDN("racfid=userid,cn=ictx") BINDPW(password))
```

The LDAPBIND class needs to be active and RACLISTed so that the LDAPHOST name is stored in the in-storage copy of the values.

The application's RACF user ID also requires READ access to the IRR.RDCEKEY profile in the FACILITY class, so that the Identity Cache can retrieve remote configuration information from RACF.

To get better performance where possible, add the APPLDATA field to the ICTX defaults profile. It should contain the IBM TDS host name given to the local Identity Cache.

Configuring z/OS sysplex for the Identity Cache

If using the Identity Cache on more than one system in a z/OS sysplex, you must ensure that RACF is enabled for sysplex communication and that the RACF database used by each member of the sysplex is synchronized. RACF recommends that all systems enabled for sysplex communication and sharing the same RACF database be members of the same sysplex. Doing this prepares you for RACF sysplex data sharing, and ensures that commands are propagated to all members of the sysplex.

For information on enabling RACF for sysplex communication, refer to *z/OS Security Server RACF General User's Guide*

Chapter 14. ICTX Java API

The ICTX Java API is the primary interface for working with the z/OS Identity Cache. The z/OS Identity Cache (described in Chapter 13, "The z/OS Identity Cache," on page 423) can be used by applications to communicate user authentication information across security domain boundaries. An application running on a z/OS or non-z/OS system can use the ICTX Java API to store identity context information in the Identity Cache on the local, or on a remote, z/OS system.

The ICTX Java API provides two logical sets of services – one for specifying and parsing authentication information, and one for storing that information in, and retrieving it from, the z/OS Identity Cache. These two logical sets of services are provided in three packages.

- The **/com/ibm/ictx/authenticationcontext** package contains interfaces and public classes for specifying and parsing user authentication information called an *authentication context*. The authenticating application uses the classes and methods provided in this package to build an authentication context for a user, and ultimately transforms it into an *identity context* object that can be stored into a z/OS Identity Cache. The terms *authentication context* and *identity context* are largely synonymous; the difference is that an identity context is just an array of bytes that can be stored in, and retrieved from, the z/OS Identity Cache. The authenticating application specifies the authentication context information using the various classes in this package, and then builds the identity context so that information can be stored in the z/OS Identity Cache. The application that retrieves the identity context can use the classes and methods of this class to parse the identity context for the authentication context information supplied by the authenticating application.
- The **/com/ibm/ictx/identitycontext** package contains interfaces and public classes for storing an identity context in, and retrieving an identity context from, the z/OS Identity Cache. Two storage mechanism classes are provided in this package for interacting with a z/OS Identity Cache. The LdapStorageMechanism class is designed for applications that will access a z/OS Identity Cache remotely using a z/OS IBM TDS server configured with ICTX extended operations, and the zOSStorageMechanism class is for applications that are accessing a z/OS Identity Cache locally. A special factory class (StorageMechanismFactory) for instantiating an object of the appropriate storage-mechanism class (based on the location of the executing code, certain configuration settings, and information supplied by the application) is also provided in this package.
- The **/com/ibm/ictx/util package** contains utility classes used by the classes in the other two packages. Classes are provided for representing an identity context and exceptions thrown by other classes.

The ICTX Java API is provided in the ictx.jar file, which is located in the `/usr/lpp/eim/lib` HFS directory.

The information provided here is only an overview of the ICTX Java API. While the interfaces and classes are listed, and the basic tasks an application can perform are summarized, detailed syntax information is not provided. For full details on the ICTX Java API, see the reference documentation provided in Javadoc format at:

`http://www.ibm.com/systems/z/os/zos/downloads/`

Configuring the IBM Tivoli Directory Server for remote services support

The ICTX Java APIs use an IBM Tivoli Directory Server client to send authentication information to the server. The ICTX extended operations support provides the database functions for this information. To enable this, the IBM TDS configuration file must have a section that identifies the ICTX extended operations support.

To enable ICTX extended operations support, do the following:

- create a new section in the IBM Tivoli Directory Server SLAPDCNF configuration file by adding the following lines:

  ```
  # ICTX extended operations support section
  plugin clientOperation ITYBIC31 ICTX_INIT "CN=ICTX"
  ```

  This statement must appear before any database definitions within the file.
- You must ensure the EIM libraries can be located via the IBM TDS LIBPATH. Add the following LIBPATH environment variable:

  ```
  LIBPATH=/usr/lib
  ```

  This statement should appear in the file specified by the ENVVAR DD in the JCL for the IBM TDS started task.

/com/ibm/ictx/authenticationcontext package

The **/com/ibm/ictx/authenticationcontext** package contains interfaces and public classes for specifying and parsing user authentication information. The authenticating application uses the classes and methods provided in this package to build an authentication context for a user, and ultimately transforms it into an IdentityContext object that can be stored into a z/OS Identity Cache. The application that retrieves the IdentityContext object can use the classes and methods of this class to parse the identity context for the information supplied by the authenticating application.

The following table lists the interfaces and classes provided in the **/com/ibm/ictx/authenticationcontext** package. For complete information on these interfaces and classes, refer to the ICTX Java API reference documentation provided in Javadoc format at:

```
http://www.ibm.com/systems/z/os/zos/downloads/
```

Table 49. Interfaces and classes in the com.ibm.ictx.authenticationcontext package

Interface	Class	Description
	ApplicationInfo	An ApplicationInfo object contains information about a particular application within the network.
AuthenticationContextBase	Type1AuthenticationContext	The AuthenticationContextBase interface represents authentication information and is implemented by the Type1AuthenticationContext class.
AuthenticationContextManagerBase	Type1AuthenticationContextManager	The AuthenticationContextManagerBase interface encapsulates methods for working with an authentication context, and is implemented by the Type1AuthenticationContextManager class.
	AuthenticationInfo	An AuthenticationInfo object contains information about an authenticated user.

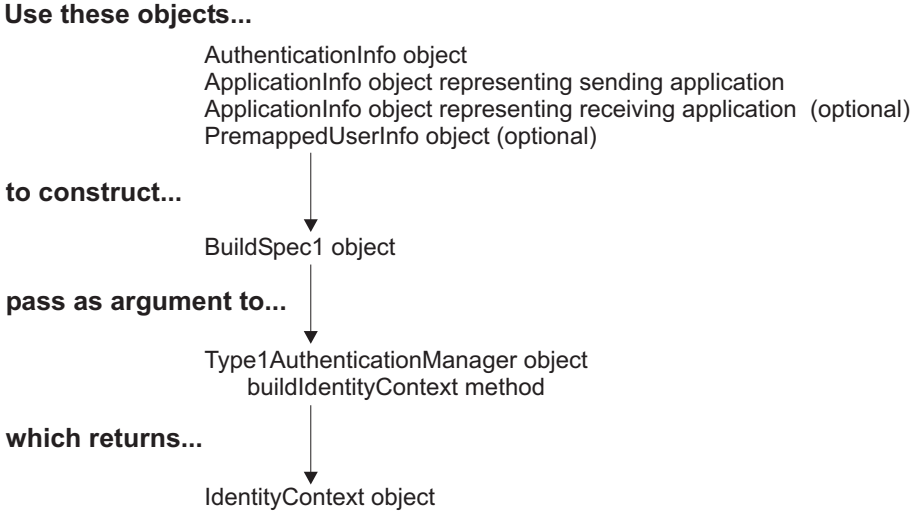
Table 49. Interfaces and classes in the com.ibm.ictx.authenticationcontext package (continued)

Interface	Class	Description
BuildSpecBase	BuildSpec1	The BuildSpecBase interface represents the information required to build an IdentityContext object, and is implemented by the BuildSpec1 class. Contains authentication, application, and z/OS mapping information.
DelegationSpecBase	DelegationSpec1	The DelegationSpecBase interface represents the information required to delegate an IndentityContext object, and is implemented by the DelegationSpec1 class. Contains application and z/OS mapping information.
	ManifestInfo	A ManifestInfo object contains manifest information that was parsed from an authentication context. An array of ManifestInfo objects (using a LIFO organization) can be obtained by the application.
	PremappedUserInfo	A PremappedUserInfo object contains information about a mapping from the authenticated user ID to a z/OS user ID.

The classes and methods for interacting with the z/OS Identity Cache to store and retrieve an IdentityContext object are organized in the **/com/ibm/ictx/identitycontext** package, while the IdentityContext class is provided in the **/com/ibm/ictx/util** package. See "/com/ibm/ictx/identitycontext package" on page 438 and "/com/ibm/ictx/util package" on page 442 for more information.

Creating an identity context object from authentication context information

The following figure illustrates the classes and methods an application can use to specify authentication context information for a user, and, from that authentication context information, build an identity context object that can stored in the z/OS Identity Cache.



To create an identity context, an application needs to:

1. construct the objects that will be passed as arguments when constructing a BuildSpec1 object. The objects that will be passed as arguments to the BuildSpec1 class constructor are:

a. an AuthenticationInfo object (authInfo argument) representing user authentication information. At a minimum, the application provides the name of the user who was authenticated and the name of the user registry for the authenticated user. The application can also supply the DNS name of the host system where the user was authenticated, the mechanism used to authenticate the user, and the security label associated with this user. In addition, implementation-specific data can be provided.

b. an ApplicationInfo object (sndAppInfo argument) that represents the current application. At a minimum, the application provides its identifier name and an instance name that uniquely identifies it in the network. In addition, implementation-specific data can be provided.

c. optionally, another ApplicationInfo object (rcvAppInfo argument) — this one identifying the application that is the intended recipient of the identity context. The application might want to provide this information so that the receiving application can verify that it is the intended recipient.

d. optionally, a PremappedUserInfo object (premapped argument) that enables the application to provide a mapping from the local authenticated user ID to a z/OS user ID valid on the system hosting the z/OS Identity Cache. Whether such mappings are required, accepted but not required, or ignored by the z/OS Identity Cache depends on how the Identity Cache is configured. See “Configuring user ID mapping” on page 427 for more information.

2. construct a BuildSpec1 object, which represents the information required to build an authentication context object, from the objects you created in the preceding step. The application can specify the AuthenticationInfo object, the ApplicationInfo object(s) and the PremappedUserInfo object as arguments to the BuildSpec1 constructor. If the application is not specifying the intended application recipient for the identity context or is not providing a user ID mapping, a null value can be specified instead of the ApplicationInfo object or the PremappedUserInfo object.

   When constructing the BuildSpec1 object, the application can also specify a timeout value representing the number of seconds before the authentication context should no longer be considered valid. This enables the receiving application to determine if the authentication context is still valid. By default, the timeout period is 3600 seconds (1 hour), but the application can set this to a shorter interval.

3. construct a Type1AuthenticationContextManager object and use its buildIdentityContext method to create the IdentityContext object. The application passes the BuildSpec1 object created in the preceding step to the buildIdentityContext method, which returns the IdentityContext object.

The IdentityContext created contains all the authentication context information provided by the application in a format that can be stored and retrieved from the z/OS Identity Cache using classes and methods provided in the **/com/ibm/ictx/identitycontext** package. The receiving application will be able to parse this IdentityContext object to get a Type1AuthenticationContext from which it can retrieve the authentication information. See “Parsing an identity context object for authentication context information” on page 436 for more information.

Once the authenticating application has created an IdentityContext object, it can store it in the z/OS Identity Cache as described in “Storing an identity context object in the z/OS Identity Cache” on page 441. Alternatively, it could forward the IdentityContext to another ICTX application to store into the Identity Cache (as described in “Delegating an identity context object” on page 435).

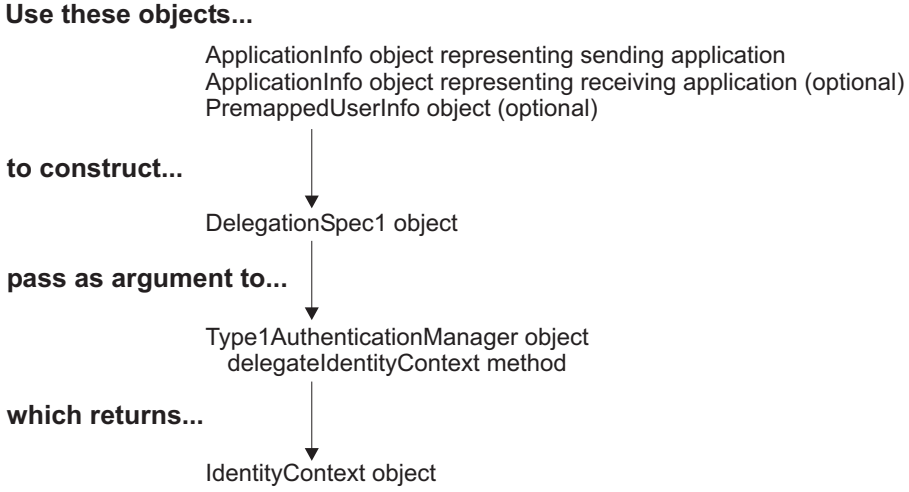
Delegating an identity context object

When authentication context information is initially generated into an IdentityContext object, a manifest entry is created. When an application passes an identity context reference to another application, that receiving application can later obtain a ManifestInfo object, and from it ascertain the application that created the authentication context, the intended receiving application (if provided by the sending application), and any user ID mapping specified by the sending application.

If the authenticating application forwards the IdentityContext object to another ICTX application, it can delegate the IdentityContext object. Delegating an IdentityContext object creates a new IdentityContext object from the first, with an additional manifest entry indicating the current application, the receiving application, and updated mapping information (if necessary). The receiving application can get an array of ManifestInfo objects from which it can ascertain the application that created the authentication context, as well as the application(s) to which it was delegated.

For example, the authenticating application could create an IdentityContext object with the authentication context information for a particular user, which it delegates to another application to perform an EIM lookup. That application delegates it to a third application, supplying a user ID mapping. The third application stores the IdentityContext object in the z/OS Identity Cache. The receiving application will be able to parse the IdentityContext object for an array of three ManifestInfo objects, from which it can identify the application that originally created the IdentityContext object, as well as each application to which it was delegated. In this way, a history of the identity context is kept within the identity context itself.

The following figure illustrates the classes and methods an application can use to delegate an identity context object to create a new identity context object with an additional manifest entry.



To delegate an identity context, an application needs to:

1. construct the objects that will be passed as arguments when constructing a DelegationSpec1 object. The objects that will be passed as arguments to the DelegationSpec1 class constructor are:
   a. an ApplicationInfo object that represents the current application. At a minimum, the application provides its identifier name and an instance name that uniquely identifies it in the network. In addition, implementation-specific data can be provided.

b. optionally, another ApplicationInfo object — this one identifying the application to which the identity context is being delegated. The application might want to provide this information so that the receiving application can verify that it is the intended recipient.

c. optionally, a PremappedUserInfo object that enables the application to provide a mapping from the local authenticated user ID to a z/OS user ID valid on the system hosting the z/OS Identity Cache. Whether such mappings are required, accepted but not required, or ignored depends on how the z/OS Identity Cache is configured. See “Configuring user ID mapping” on page 427 for more information.

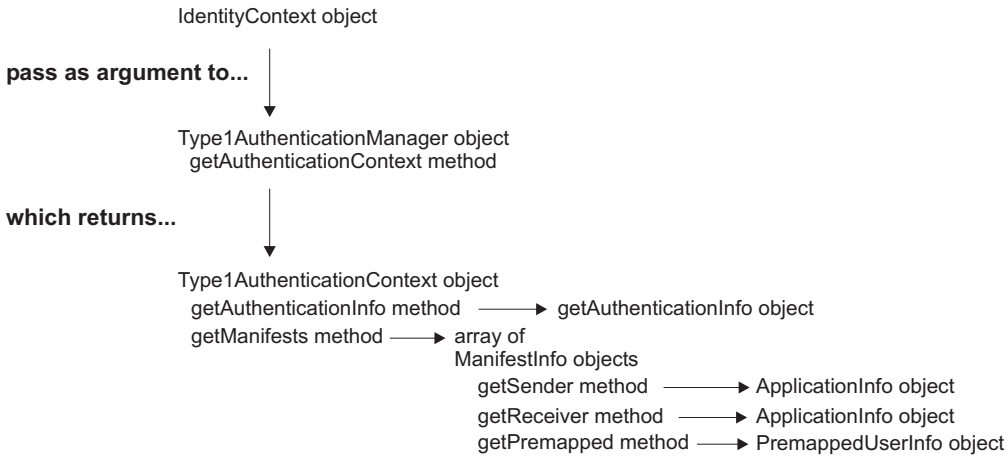
2. construct a DelegationSpec1 object, which represents the information required to delegate an identity context, from the object(s) you created in the preceding step. The application can specify the ApplicationInfo object(s) and the PremappedUserInfo object as arguments to the DelegationSpec1 constructor. If the application is not specifying the intended application recipient for the identity context or is not providing a user ID mapping, a null value can be specified instead of the ApplicationInfo object or the PremappedUserInfo object.

   When constructing the DelegationSpec1 object, the application can also specify a timeout value representing the number of seconds before the authentication context should no longer be considered valid. This enables the receiving application to determine if the authentication context is still valid. By default, the timeout period is 3600 seconds (1 hour), but the application can set this to a shorter interval.

3. use the delegateIdentityContext method of the Type1AuthenticationContextManager object to create a new IdentityContext object with an additional manifest entry. The application passes the DelegationSpec1 object created in the preceding step along with the original IdentityContext object to the delegateIdentityContext method, which returns the new IdentityContext object.

Parsing an identity context object for authentication context information

An authenticating application specifies authentication context information to create an IdentityContext object which can be stored in the z/OS Identity Cache. The IdentityContext object is a byte array that can be stored and retrieved. Once retrieved from the z/OS Identity Cache, it can be parsed by an application to examine the authentication context information. The following figure illustrates how a receiving application can parse an IdentityContext object to ascertain the authentication context information for a user.



To parse an identity context object for authentication context information, an application needs to:

1. construct a Type1AuthenticationContextManager object and use its getAuthenticationContext method to parse the identity context byte array. The application passes the IdentityContext object as an argument to the getAuthenticationContext method, which returns a Type1AuthenticationContext object.
2. from the Type1AuthenticationContext object, the applicaiton can obtain information about the authenticated user, as well as the authenticating application that built the IdentityContext object and any applications to which the IdentityContext object was delegated.

   To obtain information about the authenticated user, an application:

   a. calls the getAuthenticationInfo method of the Type1AuthenticationContext class. The getAuthenticationInfo method returns an AuthenticationInfo object.
   b. calls methods of the AuthenticationInfo class to obtain the desired information:

Table 50. Methods provided by the AuthenticationInfo class

This method:	Returns:
getUser	the name of the user who was authenticated. It may be used as the source user in a mapping lookup operation.
getRegistry	the name of the user registry for the authenticated user. This registry name may be used by a mapping lookup operation.
getMechanism	the mechanism used to authenticate the user.
getHostName	the DNS name of the host system where the user was authenticated.
getSecurityLabel	the security label associated with this user.
getImplSpecific	implementation-specific data.

   To obtain information about the authenticating application (and any applications to which the IdentityContext was delegated), the application:

   a. calls the getManifests method of the Type1AuthenticationContext class. The getManifests method returns an array of ManifestInfo objects — one ManifestInfo object recording the information supplied when the IdentityContext was created, and an additional ManifestInfo object for each time the IdentityContext was delegated. The ManifestInfo objects are in LIFO order.
   b. calls methods of the ManifestInfo class as needed to obtain information about the manifest (such as the time it was created, and how much longer it is considered valid) or objects representing the sending application, the receiving application, and any User ID mappings provided by the sending applicaiton.

Table 51. Methods provided by the ManifestInfo class

This method:	Returns:
getSender	an ApplicationInfo object representing the sending application. Using methods of this class, the current application can obtain information about the sending application. It can use: • The getIdentifier method to obtain the identifier name that has been defined for the application. • The getInstanceName method to obtain the instance name that uniquely identifies the application in the network. • The getImplSpecific method to obtain implementation-specific data.

Table 51. Methods provided by the ManifestInfo class (continued)

This method:	Returns:
getReceiver	an ApplicationInfo object representing the application information for the receiving application. It can use: • The getIdentifier method to obtain the identifier name that has been defined for the application. • The getInstanceName method to obtain the instance name that uniquely identifies the application in the network. • The getImplSpecific method to obtain implementation-specific data.
getPremapped	a PremappedUserInfo object representing a user ID mapping provided by the sending application. Using methods of this class, the current application can obtain the user mapping as well as information about the mapping operation. It can use: • The getUser method to obtain the name of the premapped target user. • The getRegistry method to obtain the name of the target user registry that was used for the mapping lookup operaton. • The getMappingQualifiers method to obtain the additional information that may have been used for the mapping operation. • The getMappingSource method to obtain the type of information was used as the source for the mapping lookup operation. • The getImplSpecific method to obtain implementation-specific data.
getCreationTime	the time that this manifest was created.
getTimeToLive	the remaining time (in seconds) that the manifest information is considered valid.
getCounter	the counter related to this manifest. Each time an authentication context is delegated, another manifest entry is added. The first manifest entry will have a counter value of 1. The top manifest entry in the array is the last manifest entry added to the authentication context.

/com/ibm/ictx/identitycontext package

The **/com/ibm/ictx/identitycontext** package contains interfaces and public classes for storing an identity context in, and retrieving an identity context from, the z/OS Identity Cache. Two storage mechanism classes are provided in this package for interacting with a z/OS Identity Cache. The LdapStorageMechanism class is designed for applications that will access a z/OS Identity Cache remotely using a z/OS IBM TDS server configured with ICTX extended operations, and the zOSStorageMechanism class is for applications that are accessing a z/OS Identity Cache locally. A special factory class (StorageMechanismFactory) for instantiating an object of the appropriate storage-mechanism class based on system configuration settings and/or information supplied by the application is also provided in this package.

The following table lists the interfaces and classes provided in the **/com/ibm/ictx/identitycontext** package. For complete information on these interfaces and classes, refer to the ICTX Java API reference documentation provided in Javadoc format at:

`http://www.ibm.com/systems/z/os/zos/downloads/`

Table 52. Interfaces and classes in the com.ibm.ictx.authenticationcontext package

Interface	Class	Description
StorageMechanism	LdapStorageMechanism	The LdapStorageMechanism class implements the StorageMechanism interface (which defines the methods required by a storage mechanism to support identity contexts). An LdapStorageMechanism object allows remote access to a z/OS Identity Cache. Remote interaction with the z/OS Identity Cache is achieved using a z/OS IBM TDS server configured with ICTX extended operations.
	zOSStorageMechanism	The zOSStorageMechanism class implements the StorageMechanism interface (which defines the methods required by a storage mechanism to support identity contexts). An zOSStorageMechanism object allows local access to the z/OS Identity Cache. Operations are performed on the local system using Java JNI. This class is only valid on z/OS systems.
	IdentityContextCredential	An IdentityContextCredential object represents a credential that enables an application to retrieve an IdentityContext object from the z/OS Identity Cache.
	StorageMechanismFactory	The StorageMechanismFactory object is a special factory object an application can use to obtain the correct type of storage mechanism object (either an LdapStorageMechanism object or a zOSStorageMechanism object) based on the location of executing code and certain configuration information.

The classes and methods for creating an IdentityContext object from authentication context information, and parsing an IdentityContext object to retrieve the authentication context information, are organized in the **/com/ibm/ictx/authenticationcontext** package. The IdentityContext class is provided in the **/com/ibm/ictx/util** package. See "/com/ibm/ictx/authenticationcontext package" on page 432 and "/com/ibm/ictx/util package" on page 442 for more information.

Creating a storage mechanism object for interacting with the z/OS Identity Cache

A storage mechanism object enables an application to interact with a local or remote z/OS Identity Cache to store and retrieve IdentityContext objects. To access the Identity Cache on a remote z/OS system, an application uses an LdapStorageMechanism object. To access an Identity Cache that is local to an application running on z/OS, an application uses a zOSStorageMechanism object.

In addition to the LdapStorageMechanism and zOSStorageMechanism class constructors, a special factory class (StorageMechanismFactory) is provided for creating the correct type of storage mechanism object based on the location of the executing code, certain configuration settings, and information provided by the application.

- The LdapStorageMechanism class constructor creates a storage mechanism object that enables an application running on a z/OS or non-z/OS system to interact with an Identity Cache on a remote z/OS system. In order to connect to the remote system, a z/OS IBM TDS server with ICTX extended operations will need to be identified, and bind credentials will need to be supplied.

  An application running on a non-z/OS system must provide this information in a hash table supplied to the LdapStorageMechanism class constructor. The hash table should include values specifying:

  - a URL for the z/OS IBM TDS server that provides remote access to the Identity Cache, such as ldap://some.big.host
  - a bind DN
  - a bind password

  For example:

```
Hashtable bindInfo = new Hashtable();
          bindInfo.put(Context.PROVIDER_URL, "ldap://myserver.com)");
          bindInfo.put(Context.SECURITY_PRINCIPAL, "racfid=user01,cn=ICTX");
          bindInfo.put(Context.SECURITY_CREDENTIALS, "password");
```

  An application running on a z/OS system can also provide this information in a hash table supplied to the LdapStorageMechanism class constructor, or it could instead supply this information as default values in an in-storage copy of the IRR.ICTX.DEFAULTS profile in the LDAPBIND class. See “Configuring the z/OS Identity Cache” on page 426 for more information.
- The zOSStorageMechanism class constructor creates a storage mechanism object that provides direct access to the z/OS Identity Cache for applications running locally.
- The getStorageMechanism method of the StorageMechanismFactory class returns either the zOSStorageMechanism or LdapStorageMechanism based on:
  - whether the application is running on a z/OS or non-z/OS system.
  - an input hash table object passed as an argument to the getStorageMechanism method specifying a URL for a z/OS IBM TDS server, a bind DN, and a bind password.
  - for z/OS applications, default connection information specified in an in-storage copy of the IRR.ICTX.DEFAULTS profile in the LDAPBIND class.

To determine which storage mechanism to run, the factory class uses the following criteria:

- If the Java application is not running on z/OS:
  - If no URL for a z/OS IBM TDS server was provided by the application, an error exception is returned.
  - If a URL is provided by the application, the application can access the remote Identity Cache, and an instance of the LdapStorageMechanism is returned.
- If the Java application is running on z/OS:
  - If no URL is provided and the in-storage copy of the IRR.ICTX.DEFAULTS profile has no URL, then access to the local Identity Cache is assumed, and an instance of the zOSStorageMechanism class is returned.
  - If a URL is provided, or the in-storage copy of the IRR.ICTX.DEFAULTS profile has a URL, that matches the value in the in-storage copy of the APPLDATA field, then access to the local Identity Cache is configured and an instance of the zOSStorageMechanism class is returned.
  - Otherwise, if the URL value from the hash table, or the in-storage copy of the IRR.ICTX.DEFAULTS profile, doesn’t match the value in the in-storage copy of the APPLDATA field, access is required to a remote Identity Cache and an instance of the LdapStorageMechanism class is returned.

The following table shows the possible scenarios for accessing the Identity Cache. Note that the StorageMechanismFactory returns a zOSStorageMechanism object when the Identity Cache is local, and an LDAPStorageMechanism object when remote.

Table 53. Identity Cache calling scenarios

Application		z/OS Setup				
Provides URL?	**Running on z/OS?**	**In-storage ICTX defaults LDAPHOST value**	**In-storage ICTX defaults APPLDATA value**	**Identity Cache access type**	**Return Code**	**Storage mechanism object returned by the factory**
null	No	n/a	n/a	–	Error	null
Yes	No	n/a	n/a	Remote	Success	LDAP
null	Yes	No data	Any	Local	Success	z/OS
null	Yes	URL	No data	Remote	Success	LDAP
null	Yes	URL	URL doesn't match LDAPHOST	Remote	Success	LDAP
null	Yes	URL	URL does match LDAPHOST	Local	Success	z/OS
Yes	Yes	Any	No data	Remote	Success	LDAP
Yes	Yes	Any	URL doesn't match parameter	Remote	Success	LDAP
Yes	Yes	Any	URL does match parameter	Local	Success	z/OS

Storing an identity context object in the z/OS Identity Cache

To store an IdentityContext object in the z/OS Identity Cache, the application uses the storeIdentityContext method of the storage mechanism object that is providing access to the Identity Cache — either an LdapStorageMechanism or zOSStorageMechanism object. The application specifies the IdentityContext object as an argument to the storeIdentityContext method.

The storeIdentityContext method returns an IdentityContextCredential object for the stored identity context. An IdentityContextCredential object represents a credential that enables an application to retrieve an IdentityContext object from the z/OS Identity Cache. An IdentityContextCredential object consists of a user name and a reference. The user name is a flag that indicates the data is stored in an Identity Cache, and the reference is what is needed to retrieve the data from the Identity Cache. The user name and reference can be returned by the getUserName and getReference methods, and then forwarded to the application that needs to retrieve the IdentityContext object.

Retrieving an identity context object from the z/OS Identity Cache

To retrieve an IdentityContext object from the z/OS Identity Cache, an application uses the retrieveIdentityContext method of the storage mechanism object that is providing access to the Identity Cache — either an LdapStorageMechanism or zOSStorageMechanism object. The application specifies an IdentityContextCredential object representing the stored identity context as an argument to the retrieveIdentityContext method, which returns the IdentityContext object.

An IdentityContextCredential object consists of a user name and a reference that will have been forwarded from the application that stored the IdentityCache object to the application that needs to retrieve it. Using the user name and reference strings, the application constructs the IdentityContextCredential object, and passes it to the retrieveIdentityContext method to obtain the IdentityContext.

/com/ibm/ictx/util package

The **/com/ibm/ictx/util** package contains utility classes used by the classes in the other two packages. Classes are provided for representing an identity context and exceptions thrown by other classes.

Table 54. Interfaces and classes in the com.ibm.ictx.authenticationcontext package

Class	Description
IdentityContext	An IdentityContext object contains the information for an identity context.
IdentityContextException	An IdentityContextException object is an exception thrown by ICTX API class.

Sample ICTX application

The following sample code illustrates the steps an application might perform to store an identity context in, and later retrieve it from, the z/OS Identity Cache. The code for both the authenticating application (which constructs the IdentityContext object and stores it in the Identity Cache), and the server application (which retrieves the IdentityContext object and parses it for the user information) is shown.

This first example shows the code for the authenticating application. It builds the IdentityContext object, creates a storage mechanism object for interacting with the z/OS Identity Cache, and stores the IdentityContext object into the Identity Cache. It then passes the user name and a reference for the IdentityContextCredential to the server application so that it can retrieve the IdentityContext object.

```
package storeretrievesample;

import java.util.Properties;

import javax.naming.directory.InitialDirContext;

import com.ibm.ictx.authenticationcontext.ApplicationInfo;
import com.ibm.ictx.authenticationcontext.AuthenticationInfo;
import com.ibm.ictx.authenticationcontext.BuildSpec1;
import com.ibm.ictx.authenticationcontext.PremappedUserInfo;
import com.ibm.ictx.authenticationcontext.Type1AuthenticationContextManager;
import com.ibm.ictx.identitycontext.IdentityContextCredential;
import com.ibm.ictx.identitycontext.StorageMechanism;
import com.ibm.ictx.identitycontext.StorageMechanismFactory;
import com.ibm.ictx.util.IdentityContext;

import storeretrievesample.RetrieveSample;

/**
 * Class: StoreSample
 *
 * This is a sample of how to store a user name that has been authenticated in
 * the z/OS Identity Cache using the ibm.com.ictx interfaces and classes.
 * Typically, an application that authenticates a user will store the
 * user information in the Identity Cache.
 *
 */
public class StoreSample {
```

```
/**
 * Method: storeUserInCache
 *
 * This is a sample method that stores the authenticated user's name in the
 * Identity Cache.
 *
 * @param userName
 *         The name of the authenticated user
 * @param userRegistry
 *         The name of the registry used for authentication. This
 *         registry name must be defined to EIM if the Identity Cache is
 *         going search EIM for a mapping from the user name to a z/OS
 *         user id.
 * @param registryType
 *         The type or kind of registry it is, i.e. RACF, LDAP, Tivoli Access Manager
 * @param location
 *         This is where the authentication event occurred in the network. It
 *         can be a DNS Host name or the name of an application instance. Something
 *         That allows an auditor to identify where the user was authenticated.
 * @return an identity context credential;
 * @throws Exception
 */
IdentityContextCredential storeUserInCache(String userName,
               String userRegistry, String registryType, String location)
               throws Exception {

       // Application constants for use with the identity context APIs
       final String NO_APPLICATION_IDENTIFIER = null;
       final String NO_APPLICATION_INSTANCE_NAME = null;
       final String NO_SECURITY_LABEL = null;
       final String NO_IMPLEMENTATION_SPECIFIC_DATA = null;

       // Create an identity context from the user information. There
       // is no application or premapped information in this example.
       AuthenticationInfo userInfo = new AuthenticationInfo(userName,
                     userRegistry, location, registryType, 1 NO_SECURITY_LABEL,
                     NO_IMPLEMENTATION_SPECIFIC_DATA);
       ApplicationInfo sendingAppInfo = null; // No sending application info
       ApplicationInfo receivingAppInfo = null; // No receiving application info
       PremappedUserInfo premappedUserInfo = null; // No premapping
       int timeoutValue = 300; // This user information is good for 5 minutes
       BuildSpec1 bldInfo = new BuildSpec1(userInfo, sendingAppInfo,
                     receivingAppInfo, premappedUserInfo, timeoutValue);
       Type1AuthenticationContextManager cm = new Type1AuthenticationContextManager();
       IdentityContext iContext = cm.buildIdentityContext(bldInfo);

       // Store the identity context in the z/OS Identity Cache
       // - Provide the bind credentials
       // - Create the instance of the storage mechanism
       // - Store the identity context getting an identity context credential in return
       Properties iCacheLocation = new Properties();
       iCacheLocation.put(InitialDirContext.PROVIDER_URL,
                     "ldap://some.zOS.ldap"); // the Cache location
       iCacheLocation.put(InitialDirContext.SECURITY_PRINCIPAL,
                     "racfid=USER01,cn=ictx"); // the bind dn
       iCacheLocation.put(InitialDirContext.SECURITY_CREDENTIALS,
                     "secret"); // the bind password

       StorageMechanismFactory smf = new StorageMechanismFactory();
       StorageMechanism sm = smf.getStorageMechanism(iCacheLocation);

       IdentityContextCredential userHandle = sm.storeIdentityContext(
                     iContext, timeoutValue);

       return userHandle;
 } // storeUserInCache
```

```
      /**
       * Method: main
       *
       * This method stores information about an authenticated
       * user in the z/OS Identity Cache and passes the Identity
       * Cache credentials to a service that only accepts a
       * user id and password.
       *
       */
      public void main() {

             String userName = "cn=Joe User,c=us";
             String userRegistry = "Some Ldap User Registry";
             String registryType = "LDAP";
             String location = "ldap://Some.AIX.Directory";

             try {
                    // Step 1: Authenticate the user

                    // Step 2: Store the authenticated user's info in the Identity Cache
                    IdentityContextCredential userh = storeUserInCache(userName,
                                  userRegistry,registryType,location);

                    // Step 3: Pass the identity context userid and reference
                    // to the service that normally accepts a userID and password
                    RetrieveSample rs = new RetrieveSample();
                    rs.sampleService(userh.getUserName(), userh.getReference());
             } catch (Exception e) {
                    System.out.println("Exception Occurred: " + e);
             }
       } // main
```

This next example shows the code for the server application. It constructs an IdentityContextCredential object using the user name and reference string passed by the authenticating application. It then creates a storage mechanism object for interacting with the z/OS Identity Cache, and retrieves the IdentityContext object. It then parses this object to get the user authentication information.

```
package storeretrievesample;

import java.util.Properties;

import javax.naming.directory.InitialDirContext;

import com.ibm.ictx.authenticationcontext.AuthenticationInfo;
import com.ibm.ictx.authenticationcontext.Type1AuthenticationContext;
import com.ibm.ictx.authenticationcontext.Type1AuthenticationContextManager;
import com.ibm.ictx.identitycontext.IdentityContextCredential;
import com.ibm.ictx.identitycontext.StorageMechanism;
import com.ibm.ictx.identitycontext.StorageMechanismFactory;
import com.ibm.ictx.util.IdentityContext;

/**
 * Class: RetrieveSample
 *
 * This Java class retrieves the identity context stored by
 * the StoreSample class in the z/OS Identity Cache.
 */

public class RetrieveSample {


       /**
        * Method: getUserInfoFromCache
        *
        * This method accepts the userID and reference from an identity
        * identity context credential and retrieves the identity context
```

```
 * from the z/OS Identity Cache.
 *
 * @param userId the userId portion of an identity context credential.
 * @param userPassword the reference portion of an identity context credential.
 * @return an identity context
 * @throws Exception
 */
IdentityContext getUserInfoFromCache(String userId, String reference) throws Exception {

       Properties icacheLocation = new Properties();
       icacheLocation.put(InitialDirContext.PROVIDER_URL,
                      "ldap://some.zOS.ldap"); // the Identity Cache location
       icacheLocation.put(InitialDirContext.SECURITY_PRINCIPAL,
                      "racfid=USER02,cn=ictx"); // the bind dn
       icacheLocation.put(InitialDirContext.SECURITY_CREDENTIALS,
                      "secret"); // the bind password

       StorageMechanismFactory smf = new StorageMechanismFactory();
       StorageMechanism sm = smf.getStorageMechanism(icacheLocation);

       IdentityContextCredential iCtxCred =
              new IdentityContextCredential(userId,reference);

       return sm.retrieveIdentityContext(iCtxCred);
}

/**
 * Method: findLocalUserId
 *
 * This method performs searchs an EIM domain for the local user id
 * given the userName and registry from an identity context.
 *
 * @param iCtx
 * @return local user id or null
 * @throws Exception
 */
String findLocalUserId( IdentityContext iCtx)throws Exception {

       // Step 1. Extract the user name and registry from the
       // identity context.
       Type1AuthenticationContextManager acMgr = new Type1AuthenticationContextManager();
       Type1AuthenticationContext ac = (Type1AuthenticationContext)
       acMgr.getAuthenticationContext(iCtx);
       AuthenticationInfo ai = ac.getAuthenticationInfo();
       String sourceUser = ai.getUser();
       String sourceRegistry = ai.getRegistry();

       // Step 2. Contact the mapping service to retrieve the
       // local user ID for the authenticated user.
       return "USER01";
 }

 public void sampleService(String userId, String password ) throws Exception {

        String userName = null;

        // Step 1. Authenticate the user
        if (IdentityContextCredential.isIctxUserName(userId)) {
                // Retrieve the identity context for the identity context
                // reference from the Identity Cache
                IdentityContext iCtx = getUserInfoFromCache(userId,password);
                // Call a mapping service to get the local user id for this user.
                userName = findLocalUserId(iCtx);
        } else {
                // This is the authentication check for regular
                // user IDs. Assumes it worked.
                userName = userId;
```

```
            }
            // Step 2. Assert the user identity
            // Step 3. perform the service
            } // sampleService

}
```

Chapter 15. Accessing RACF remotely to perform authorization checks and create audit records

Remote authorization and auditing allows resource managers that do not reside on z/OS to centralize authorization decisions and security event logging using z/OS Security Server RACF through the IBM TDS server. These services are provided through LDAP extended operations. The requests come in the form of a DER-encoding of the ASN.1 syntax. The following sections provide details on these remote authorization and auditing requests.

Because there are many mechanisms available for user authentication and mapping, the remote services do not provide an authentication option. Applications that use the authorization service are responsible for authenticating users. Applications that expect users to provide a RACF user ID and password can authenticate by performing a simple bind to the SDBM component of the IBM Tivoli Directory Server. For details on how to bind to SDBM refer to *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191. If mapping from a non-RACF user ID to a RACF user ID is to be performed during authentication, another option is to use the native authentication feature of the LDBM component of the IBM TDS LDAP server. Configuration and use of the IBM TDS native authentication feature is also documented in *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191.

Note that this information describes remote authorization and audit specifically. For full details on auditing controls in z/OS, see *z/OS Security Server RACF Auditor's Guide*.

Using remote authorization and audit

The remote authorization and audit services are enabled when the ICTX extended operations component is configured for IBM Tivoli Directory Server. Refer to "Configuring the IBM Tivoli Directory Server for remote services support" on page 432 for instructions.

An application or resource manager that uses the remote audit or authorization LDAP extended operation must be capable of generating a request, sending it through the network to the appropriate z/OS IBM TDS server, and interpreting the response from the z/OS IBM TDS server. The following steps represent the typical sequence of events that are specific to the LDAP extended operations for the remote authorization and auditing:

1. The application must perform a simple bind to the server using an authorized `racfid=`*`userid`*`,cn=ictx` bind distinguished name.
2. The application must build a DER-encoded extended operation request having the defined ASN.1 syntax that is specific to the audit or authorization request. That request can then be included with the z/OS IBM TDS server handle and specific request OID on the LDAP client call, such as **ldap_extended_operation_s()**, to build the LDAP message and send it to the server.
3. The z/OS IBM TDS receives the request and routes it to the ICTX component, where it is decoded and processed. ICTX verifies the correct syntax and the requestor's authority before invoking the SAF authorization check or audit service to satisfy the request. The result of the SAF service is a DER-encoded response that LDAP returns.

4. The application must decode the response in order to interpret the results. A nonzero LdapResult code indicates the request was not processed by the ICTX component. A nonzero LdapResult is accompanied by a reason string in the response that may provide additional diagnostic information.

   Note: A zero LdapResult code does not necessarily imply the request was processed successfully (or for authorization, that a user has the specified access). It does, however, indicate that an extended operation ResponseValue was returned. The application should verify that the ICTX ResponseCode within the ResponseValue indicates success (0). A nonzero ResponseCode indicates one or more request items resulted in errors (or unauthorized users). The application should check the MajorCode within each response item to determine which returned failures. The application should be aware that ICTX may not return a response item corresponding to each request item in the event of a severe error, such as an error encountered in the DER encoding.

The application may send as many requests as needed throughout a single bound session, and should unbind from IBM TDS when it has finished processing ICTX requests.

Profile authorizations for working with remote services

In order to recognize that a client is authorized to use remote services, the remote service requestor or client is required to have specific RACF authorization. User distinguished names with the suffix **cn=ictx** bind through the z/OS IBM TDS ICTX component. These names have the following format:

```
racfid=userid,cn=ictx
```

The password for the distinguished name must be the RACF user ID's password and this password is validated by RACF. Bind distinguished names are authenticated using native authentication. Only simple bind is supported. Kerberos and SSL binds are not supported.

Once accepted as a valid request, the bind user's authorization to use the services is verified. For remote authorization, the user must have at least READ access to FACILITY class profile IRR.LDAP.REMOTE.AUTH in order to check the user's own access to a resource. To check another user's access. the user must have at least UPDATE access to FACILITY class profile IRR.LDAP.REMOTE.AUTH. For example:

```
RDEFINE FACILITY IRR.LDAP.REMOTE.AUTH UACC(NONE)

PERMIT IRR.LDAP.REMOTE.AUTH CLASS(FACILITY) ID(BINDUSER)
ACCESS(UPDATE)
```

For remote auditing requests, the bind user must have at least READ access to FACILITY class profile IRR.LDAP.REMOTE.AUDIT.

```
RDEFINE FACILITY IRR.LDAP.REMOTE.AUDIT UACC(NONE)

PERMIT IRR.LDAP.REMOTE.AUDIT CLASS(FACILITY) ID(BINDUSER)
ACCESS(READ)
```

Authority to use the remote services is determined when the user binds through ICTX.

The remote audit operation calls the SAF service R_auditx. The service runs under the identity of the IBM TDS server, which must have the appropriate service

authorization. The user ID associated with the IBM TDS server must be granted at least READ access to FACILITY class profile IRR.RAUDITX.

```
RDEFINE FACILITY IRR.RAUDITX UACC(NONE)

PERMIT IRR.RAUDITX CLASSS(FACILITY) ID(LDAPSRV) ACCESS(READ)
```

Remote authorization requests

The remote authorization request must contain the DER-encoding of the ASN.1 syntax. The following is the remote authorization request syntax:

```
Request OID:  1.3.18.0.2.12.66

RequestValue ::= SEQUENCE  {
RequestVersion INTEGER,
ItemList  SEQUENCE of
  Item    SEQUENCE {
   ItemVersion INTEGER,
   ItemTag INTEGER,
   UserOrGroup IA5String,
   Resource IA5String,
   Class    IA5String,
   Access   INTEGER,
   LogString IA5String
  }}
```

Where:

RequestValue
The name for the entire sequence of authorization request data.

RequestVersion
The format of the request value. Version 1 indicates a user authorization request; each individual Item in the ItemList will be an authorization request for a RACF userid. Version 2 indicates a user authorization or a group authorization request; each individual Item in the ItemList will be an authorization request for either a RACF userid or a RACF group.

ItemList
A sequence of one or more items, which allows for multiple authorization checks within a single ICTX request. The size of the entire encoded RequestValue should be limited to 16 million bytes unless your encoding routine or LDAP client imposes a stricter limit. If RequestVersion is 2, the ItemList can be a mixture of user authorization and group authorization items.

Item A sequence of data that represents a single authorization check.

ItemVersion
The format of the individual item. Version 1 indicates an authorization request for a RACF userid. Version 2 indicates an authorization request for a RACF group.

ItemTag
An integer set by the client for each request item and echoed in each response item. Its purpose is to assist the client in correlating multiple request responses, and has no influence on the authorization logic or logging.

UserOrGroup
If ItemVersion is 1, a RACF userid whose authority is being

checked. Its length cannot exceed 8 characters. If the length is zero, the user value defaults to the user ID associated with the bind user.

If ItemVersion is 2, a RACF group ID whose authority is being checked. Its length must be from 1 and 8 characters. Optimizations used when performing a userid authorization check are not available when performing a group ID authorization check. For this reason, it is likely that group authorization checks will execute more slowly than userid authorization checks.

Resource
A name to be matched against a RACF profile for authorization checking. The string may not include blank characters. Its length may be from 1 to the maximum RACF profile length defined for the specified class.

Class A defined RACF general resource class. It cannot be DATASET, USER, or GROUP. Its length may be from 1 to 8 characters.

Access
The level of authority requested. It must be one of the following integer values:

```
X'01' READ
X'02' UPDATE
X'03' CONTROL
X'04' ALTER
```

LogString
Any character data from 0 to 200 characters in length. It is appended to an ICTX-defined string in the SMF log record.

The following is the ASN.1 syntax for the remote authorization response message:

```
Response OID:  1.3.18.0.2.12.67

ResponseValue ::= SEQUENCE {
ResponseVersion   INTEGER,
ResponseCode  INTEGER,
ItemList   SEQUENCE of
  Item  SEQUENCE{
   ItemVersion INTEGER,
   ItemTag INTEGER,
   MajorCode   INTEGER,
   MinorCode1  INTEGER,
   MinorCode2  INTEGER,
   MinorCode3  INTEGER
  }}
```

Where:

ResponseValue
The name for the entire sequence of authorization response data.

ResponseVersion
The format of the response value. Version 1 is the only supported format.

ResponseCode
The greatest error encountered while processing the request. See Table 55 on page 451 for more details on supported ResponseCodes.

ItemList
A sequence of one or more items, which allows for multiple authorization results within a single ICTX response.

Item A sequence of data that represents the results from a single authorization check.

ItemVersion
The format of the individual item. Version 1 is the only supported format.

ItemTag
An integer echoed from the corresponding request ItemTag. The purpose of the ItemTag is to assist the client in correlating multiple request responses. ItemTag has no influence on the authorization logic or logging.

MajorCode
An integer value representing the result of the authorization check. See Table 56 for more details on error major codes.

MinorCode1
Additional details about the error. See Table 57 on page 453 for more details on error minor codes.

MinorCode2
Additional details about the error.

MinorCode3
Additional details about the error.

Remote authorization ResponseCodes

Use the following table to understand the response codes generated from the remote authorization processing. The ResponseCode represents the greatest error encountered. You may experience situations in which a request item generates an error that is not reflected in the ResponseCode, because that value is overridden by a higher-severity error.

Table 55. Remote authorization ResponseCodes

ResponseCode (decimal)	Meaning
0	All request items were processed successfully
28	Empty item list. No items are found within the ItemList sequence of the extended operation request, so no response items are returned.
61-70	The specified RequestVersion is not supported. Subtract 60 from the value to determine the highest RequestVersion that the server supports. ResponseCode 61 indicates the server supports version 1 requests only. ResponseCode 62 indicates the highest supported request level is 2.
other	Errors or warnings encountered while processing one or more request items. The value represents the highest MajorCode in the set of all response items. Verify the major and minor codes returned for each item.

Table 56. Remote authorization MajorCodes

MajorCode (decimal)	Meaning	Comment
0	Authorized	The user has the requested access to the resource.

Table 56. Remote authorization MajorCodes (continued)

MajorCode (decimal)	Meaning	Comment
2	Warning mode	The user has the requested access because warning mode is enabled for the resource. Warning mode is a feature of RACF that allows installations to try out security policies. Installations can define a profile with the WARNING attribute. When RACF performs an authorization check using the profile, it will log the event (if there are audit settings) and allow the authorization check to pass successfully. The log records can be monitored to ensure the new policy is operating as expected before putting the policy into production by turning off the WARNING attribute.
4	Undetermined	No decision could be made. The specified resource is not protected by RACF, or RACF is not installed.
8	Unauthorized	The user does not have the requested access to the resource.
12	RACROUTE error	The RACROUTE REQUEST=AUTH service returned an unexpected error. Compare the returned minor codes with the SAF & RACF codes documented in Security Server RACROUTE Macro Reference.
14	initACEE error	The initACEE callable service returned an unexpected error. Compare the returned minor codes with the SAF & RACF codes documented in Security Server RACF Callable Services.
16	Request value error	A value specified in the extended operation request is incorrect or unsupported. Check the returned minor codes to narrow the reason.
20	Request encoding error	A decoding error was encountered indicating the extended operation request contains non-compliant DER encoding, or does not match the documented ASN.1 syntax.
24	Insufficient authority	The requestor does not have sufficient authority for the requested function. The userid associated with the LDAP bind user must have the appropriate access to the FACILITY class profile IRR.LDAP.REMOTE.AUTH.
100	Internal error	An internal error was encountered within the ICTX component.

Table 57. Remote authorization MinorCodes

MinorCode (decimal)	MinorCode Meaning
0-14	MinorCode1- the SAF return code MinorCode2 - the RACF return code MinorCode3 - the RACF reason code
16-20	MinorCode1 is the extended operation request parameter number within the item. 0 - Item sequence 1 - ItemVersion 2 - ItemTag 3 - User 4 - Resource 5 - Class 6 - Access 7 - LogString
	MinorCode2 value indicates one of the following: 32 - incorrect length 36 - incorrect value 40 - encoding error
	MinorCode3 has no defined meaning.
24-100	MinorCodes1-3 have no defined meaning.

Remote authorization audit controls

The auditor can specify whether to log access attempts based on user, class, resource, or any criteria as described in *z/OS Security Server RACF Auditor's Guide*. The SMF type 80 records generated can be unloaded by using the IRRADU00 utility.

Remote auditing requests

As with remote authorization, remote auditing also uses the DER-encoded ASN.1 syntax. The remote audit request is displayed as the following:

```
Request OID:  1.3.18.0.2.12.68

RequestValue ::= SEQUENCE   {
RequestVersion INTEGER,
ItemList SEQUENCE of
  Item SEQUENCE {
   ItemVersion INTEGER,
   ItemTag INTEGER
   LinkValue OCTET STRING SIZE(8),
   Violation BOOLEAN,
   Event INTEGER,
   Qualifier INTEGER,
   Class IA5String,
   Resource IA5String,
   Logstring IA5String,
   DataFieldList SEQUENCE of
```

```
    DataField SEQUENCE{
      Type INTEGER,
      Value IA5String
  }}}
```

Where:

RequestValue
The name for the entire sequence of the audit request data.

RequestVersion
The format of the request value. Version 1 is the only currently supported format.

ItemList
A sequence of one or more items, allowing multiple audit records to be written with a single ICTX request. IBM recommends limiting the size of the entire encoded RequestValue to 16 million bytes; however, your encoding routine or LDAP client may impose a stricter limit.

Item A sequence of data that represents a single audit record.

ItemVersion
The format of the individual item. Version 1 is the only currently supported format.

ItemTag
An integer set by the client for each request item and echoed in each response item. Its purpose is to assist the client in correlating multiple request responses. The ItemTag value does not influence the audit processing, and does not appear in the audit record.

LinkValue
8 bytes of data used to mark related audit records. Specify 8 bytes of zero (X'00') if no such marking is needed.

Violation
A boolean value that indicates whether the event represents a violation (nonzero ~ TRUE) or not (zero ~ FALSE). The value is used in the R_auditx logging decision.

Event An integer 1 to 7 that identifies the security event type. The possible values are:

1 Authentication

2 Authorization

3 Authorization Mapping

4 Key Management

5 Policy Management

6 Administrator Configuration

7 Administrator Action

Qualifier
An integer 0 to 3 that describes the event result. The possible values are:

0 Success

1 Information

2 Warning

3 Failure

Class A defined RACF general resource class that may be used for audit logging determination. It cannot be DATASET, USER, or GROUP. Its length may be from 0 to 8 characters.

Resource
A name that may be matched against a RACF profile in the specified class for audit logging determination. Its length may be from 0 to 246 characters.

LogString
Any character data from 0 to 200 characters in length. It is appended to an ICTX-defined string in the SMF log record.

DataFieldList
A sequence of type/value pairs that will be logged as SMF relocates. Any number of relocates may be included, but the R_auditx service limits the total amount of this relocate data to 20 kilobytes per record.

DataField
A sequence of data that represents a single relocate section in an audit record.

Type An integer 100 to 114 corresponding to a defined relocate number. The possible values are:

100 SAF identifier for bind user

101 Requestor's bind user identifier

102 Originating security domain

103 Originating registry / realm

104 Originating user name

105 Mapped security domain

106 Mapped registry / realm

107 Mapped user name

108 Operation performed

109 Mechanism / object name

110 Method / function used

111 Key / certificate name

112 Caller subject initiating security event

113 Date and time security event occurred

114 Application specific data

Value Character data of the associated type that is included in the audit record.

The remote audit response is displayed as the following:

```
Response OID: 1.3.18.0.2.12.69

ResponseValue ::= SEQUENCE {
ResponseVersion INTEGER,
ResponseCode INTEGER,
ItemList SEQUENCE of
```

```
Item SEQUENCE{
 ItemVersion INTEGER,
 ItemTag INTEGER,
 MajorCode INTEGER,
 MinorCode1 INTEGER,
 MinorCode2 INTEGER,
 MinorCode3 INTEGER
}}
```

Where:

ResponseValue
The name for the entire sequence of audit response data.

ResponseVersion
The format of the response value. Version 1 is the only supported format.

ResponseCode
The greatest error encountered while processing the request. See Table 58 for more details on supported ResponseCodes.

ItemList
A sequence of one or more items, which allows for multiple audit results within a single ICTX response.

Item A sequence of data that represents a single audit request.

ItemVersion
The format of the individual item. Version 1 is the only supported format.

ItemTag
An integer echoed from the corresponding request ItemTag. The purpose of the ItemTag is to assist the client in correlating multiple request responses. The ItemTag does not influence the audit processing, and does not appear in the audit record.

MajorCode
An integer value representing the result of the audit request. See Table 59 on page 457 for more details on error major codes.

MinorCode1
Additional details about the error. See Table 60 on page 458 for more details on error minor codes.

Minor Code2
Additional details about the error.

Minor Code3
Additional details about the error.

Remote auditing response codes

Use the following table to understand the response codes generated from the remote auditing processing. The ResponseCode represents the greatest error encountered. You may experience situations in which a request item generates an error that is not reflected in the ResponseCode, because that value is overridden by a higher-severity error.

Table 58. Remote auditing ResponseCodes

ResponseCode (decimal)	Meaning
0	All request items were processed successfully.

Table 58. Remote auditing ResponseCodes (continued)

ResponseCode (decimal)	Meaning
28	Empty item list. No items are found within the ItemList sequence of the extended operation request, so no response items are returned.
61-70	The specified RequestVersion is not supported. Subtract 60 from the value to determine the highest RequestVersion that the server supports. ResponseCode 61 indicates the server supports version 1 requests only.
other	Errors or warnings encountered while processing one or more request items. The value represents the highest MajorCode in the set of all response items. Verify the major and minor codes returned for each item.

Table 59. Remote auditing MajorCodes

MajorCode (decimal)	Meaning	Comment
0	Success	The event is logged successfully.
2	Warning mode	The event is is logged, and warning mode is set for the specified resource. Warning mode is a feature of RACF that allows installations to try out security policies. Installations can define a profile with the WARNING attribute. When RACF performs an authorization check using the profile, it will log the event (if there are audit settings) and allow the authorization check to pass successfully. The log records can be monitored to ensure the new policy is operating as expected before putting the policy into production by turning off the WARNING attribute. A remote client resource manager using the remote audit service may simulate RACF warning mode logic after submitting an audit request for a failing authorization event. If the MajorCode in the response item indicates the matching resource profile has the warning mode set, the remote client resource manager may allow the check to pass successfully.
3	Logging not required	The event is not logged because no audit controls are set to require it.

Table 59. Remote auditing MajorCodes (continued)

MajorCode (decimal)	Meaning	Comment
4	Undetermined	The event is not logged. The conditions suggested by the following MinorCode combinations may or may not be intentional administrator settings: **4,0,0** - RACF is not installed or not active **8,8,8** - UAUDIT is not set, and class is not active or not RACLISTed **8,8,12** - UAUDIT is not set, class is active and RACLISTed, and a covering resource profile is not found
8	Unauthorized	The user does not have authority the R_auditx service. The userid associated with the LDAP server must have at least READ access to the FACILITY class profile IRR.RAUDITX.
12	R_auditx error	The R_auditx service returned an unexpected error. Compare the returned minor codes with the SAF & RACF codes documented in *z/OS Security Server RACF Callable Services*.
16	Request value error	A value specified in the extended operation request is incorrect or unsupported. Check the returned minor codes to narrow the reason.
20	Request encoding error	A decoding error was encountered indicating the extended operation request contains non-compliant DER encoding, or does not match the documented ASN.1 syntax.
24	Insufficient authority	The requestor does not have sufficient authority for the requested function. The userid associated with the LDAP bind user must have at least READ access to the FACILITY class profile IRR.LDAP.REMOTE.AUDIT.
100	Internal error	An internal error was encountered within the ICTX component.

Table 60. Remote auditing MinorCodes

MinorCode (decimal)	MinorCode Meaning
0-12	MinorCode1- the SAF return code MinorCode2 - the RACF return code MinorCode3 - the RACF reason code

Table 60. Remote auditing MinorCodes (continued)

MinorCode (decimal)	MinorCode Meaning
16-20	MinorCode1 is the extended operation request parameter number within the item. 0 - Item sequence 1 - ItemVersion 2 - ItemTag 3 - LinkValue 4 - Violation 5 - Event 6 - Qualifier 7 - Class 8 - Resource 9 - Logstring 10 - DataFieldList sequence 11 - DataField sequence 12 - Type 13 - Value
	MinorCode2 value indicates one of the following: 32 - incorrect length 36 - incorrect value 40 - encoding error
	MinorCode3 has no defined meaning.
24-100	MinorCodes1-3 have no defined meaning.

Remote audit controls

The remote audit service uses the R_auditx callable service documented in *z/OS Security Server RACF Callable Services* to generate SMF type 83 (subtype 4) audit records. The IRRADU00 utility can then be used to unload the generated SMF type 83 subtype 4 records. Whether or not the R_auditx service actually writes an audit record for an event, however, depends on the RACF audit controls. If the audit controls do not direct RACF to log an event that was specified in a remote audit request, the R_auditx service will not generate an audit record. If the remote application sends a remote audit request for an operation that has not been configured to be logged, this will be reflected in the remote audit response (a MajorCode of 3 indicates that the event was not logged because it is not required).

There are several ways the RACF auditor can enable logging of events from remote audit requests. For example, because the remote application will have used a RACF user ID to authenticate with z/OS through an LDAP bind operation, the auditor can enable logging for all remote audit events by setting UAUDIT for that RACF user ID.

If the remote application has specified a Class and Resource in the remote auditing requests, the auditor can enable logging for the class (using the SETROPTS LOGOPTIONS command) or the resource (using the ADDSD AUDIT, ALTDSD AUDIT, or ALTDSD GLOBALAUDIT commands). To do this, the auditor must know the class and resource specified by the application submitting the remote audit requests.

An effective way to gain granular control over which remote application events are logged, is to use the RACF dynamic class descriptor table (dynamic CDT) to define custom classes to represent specific remote applications. Once you have defined a custom class to represent a remote application, you can create resource profiles in the class to represent specific user operations that the application supports. Then, by manipulating the auditing options for the class and profiles, the RACF auditor can determine the type of information logged. For example, imagine a travel application that runs remotely and supports user operations to:

- book a flight
- cancel a flight booking
- check seat availability
- view flight information

A new custom class, @FLIGHTS is created in the dynamic CDT to represent this remote application, and profiles BOOK, CANCEL, SEATCHECK, and VIEW are created in this new class to represent the user operations that are supported. A remote audit request Item will be sent by the remote application for each user operation, and the Class and Resource parameters for each Item will identify the travel application and the particular operation. The Class and Resource parameters are used on the remote audit requests for logging determination on the z/OS server. So even though the remote audit record will be sent for each operation, whether these events are actually logged will depend on how auditing is configured for the @FLIGHTS class and the BOOK, CANCEL, SEATCHECK, and VIEW profiles. This gives the RACF auditor granular control over which user operations are logged. Configuration of the audit settings on the profiles enables the RACF auditor to, for example, log all BOOK and CANCEL requests while ignoring VIEW and SEATCHECK requests.

SMF Record Type 83 subtype 4 records

The remote audit service logs events as SMF Type 83 subtype 4 records that can be unloaded using the IRRADU00 utility. Each logged event has a unique event code with a corresponding event code qualifier, or value that indicates if the event succeeded, resulted in warning or failure, or was simply logging event information. The event codes are described in the following table:

Table 61. Remote audit event codes

Event	Command / Service
1	Authentication
2	Authorization
3	Authorization mapping
4	Key management
5	Policy management
6	Administrator configuration
7	Administrator action

The following table describes the event code qualifiers:

Table 62. Remote audit event code qualifiers

(Common) Event Code Qualifier Dec (Hex)	Description	(Common) Relocate type sections
0	Successful request or authorization.	Common relocates, 100-114
1	Event information.	
2	Not a failure, but may warrant investigation. For authorization event, grace period may be in effect.	
3	Unsuccessful request; unauthorized.	

The following are the remote audit specific extended relocates:

Table 63. Event-specific fields for remote audit events

Relocate	XML Tag	DB2 Field Name	Type	Length	Position		Comments
					Start	End	
100	localUser	SAF_LOCAL_USER	Char	8	3000	3007	SAF identifier for bind user
101	bindUser	SAF_BIND_USER	Char	256	3010	3265	Requestor's bind user identifier
102	domain	SAF_DOMAIN	Char	512	3268	3779	Originating security domain
103	regName	SAF_REG_NAME	Char	256	3782	4037	Originating registry / real m
104	regUser	SAF_REG_USER	Char	256	4040	4295	Originating user name
105	mapDomain	SAF_MAP_DOMAIN	Char	512	4298	4809	Mapped security domain
106	mapRegName	SAF_MAP_REG_NAME	Char	256	4812	5067	Mapped registry / realm
107	mapRegUser	SAF_MAP_REG_USER	Char	256	5070	5325	Mapped user name
108	action	SAF_ACTION	Char	64	5328	5391	Operation performed
109	object	SAF_OBJECT	Char	64	5394	5457	Mechanism / object name
110	method	SAF_METHOD	Char	64	5460	5523	Method / function used
111	key	SAF_KEY	Char	256	5526	5781	Key / certificate name
112	subjectName	SAF_SUBJECT_NAME	Char	256	5784	6039	Caller subject initiating security event
113	dateTime	SAF_DATE_TIME	Char	32	6042	6073	Date and time security event occurred

Table 63. Event-specific fields for remote audit events (continued)

Relocate	XML Tag	DB2 Field Name	Type	Length	Position		Comments
					Start	End	
114	otherData	SAF_OTHER_DATA	Char	2048	6076	8123	Application specific data

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:

IBM Corporation
Mail Station P300
2455 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.

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

Programming interface information

This document primarily documents information that is NOT intended to be used as Programming Interfaces of EIM.

This document also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of EIM. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Programming Interface information

The EIM APIs are a programming Interface. They are intended for customers to use in customer-written programs.

End of Programming Interface information

Trademarks

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

AIX
BookManager
DB2
eServer
iSeries
IBM
IBMLink
Language Environment
Library Reader
MVS
OS/390
OS/400
pSeries

RACF
Redbooks
Resource Link
S/390
SecureWay
TalkLink
VSE/ESA
WebSphere
xSeries
z/OS
zSeries
z/VM

Adobe Acrobat is a trademark of Adobe Systems Incorporated in the United States, other countries, or both.

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

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Tivoli is a trademark of International Business Machines Corporation or Tivoli Systems, Inc. in the United States, other countries, or both.

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

Other company, product, and service names can be trademarks or service marks for other companies.

Bibliography

The following lists titles and numbers of documents referenced in this publication.

- *z/OS XL C/C++ Run-Time Library Reference*, SA22-7821
- *IBM Tivoli Directory Server Client Programming for z/OS*, SA23-2214
- *z/OS Integrated Security Services LDAP Server Administration and Use*, SC24-5923
- *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191
- *z/OS Security Server RACF Callable Services*, SA22-7691
- *z/OS Security Server RACF Command Language Reference*, SA22-7687
- *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683
- *z/OS TSO/E REXX Reference*, SA22-7790
- *z/OS UNIX System Services Command Reference*, SA22-7802
- *z/OS UNIX System Services Planning*, GA22-7800
- *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926
- *z/OS Integrated Security Services Network Authentication Service Programming*, SC24-5927
- *z/OS Cryptographic Services System SSL Programming*, SC24-5901

Index

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

Z

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

z/OS
Integrated Security Services
Enterprise Identity Mapping (EIM)
Guide and Reference

Publication No. SA22-7875-07

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

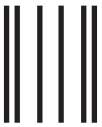
If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
SA22-7875-07

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
MHVRCFS, Mail Station P181
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

SA22-7875-07