z/OS

IBM

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

SA22-7875-01

z/OS

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

SA22-7875-01

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 289.

Second Edition, March 2004

This edition applies to Version 1 Release 5 of z/OS (5694-A01), Version 1 Release 5 of z/OS.e (5655-G52), 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
Department 55JA, Mail Station P384
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/servers/eserver/zseries/zos/webqs.html

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

Make sure to include the following in your comment or note:
- Title and order number of this 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) and z/OS.e (5655-G52). 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 1.

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 27.) 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 21 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 1.

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 and DVD collections. The CD-ROM online library collections include the IBM Softcopy Reader, which is a program that enables you to view the softcopy books.

SK3T-4269	*z/OS Version 1 Release 4 Collection* This CD-ROM collection contains the set of unlicensed books 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 CD-ROM softcopy collection 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 4 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 eserver zSeries Redbooks. The documents are provided in both BookManager and PDF formats when available.

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM® messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from the following locations to find IBM message explanations for z/OS® elements and features, z/VM®, and VSE:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e® systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX® System Services running OMVS).
- Your Windows® workstation. You can install code to access IBM message explanations on the *z/OS Collection* (SK3T-4269), using LookAt from a Windows DOS command line.
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

You can obtain code to install LookAt on your host system or Windows workstation from a disk on your *z/OS Collection* (SK3T-4269), or from the LookAt Web site (click **Download**, and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code. [1]

1. z/OS.e customers received a Memo to Licensees, (GI10-0684) that includes this key code.

To obtain your IBM Resource Link user ID and password, log on to:

`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

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*

  For more iSeries information on Enterprise Identity Mapping, see the iSeries Information Center at the following URL:

  `http://www.ibm.com/eserver/iseries/infocenter`

  iSeries-specific EIM information can be located by selecting ″Security″, and then ″Enterprise Identity Mapping″.
- *pSeries*
- *xSeries*
- *zSeries*

For more information, refer to the eServer Information Center:

`http://submit.boulder.ibm.com/eserver/`

Specific EIM information can be located by selecting ″Security″, and then ″Enterprise Identity Mapping″.

The following additional 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/servers/eserver/zseries/zos/bkserv/`
- **Redbooks**

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

`http://www.ibm.com/redbooks/`

- **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/servers/eserver/zseries/zos/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 SA22-7875-01
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.

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 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.

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

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.

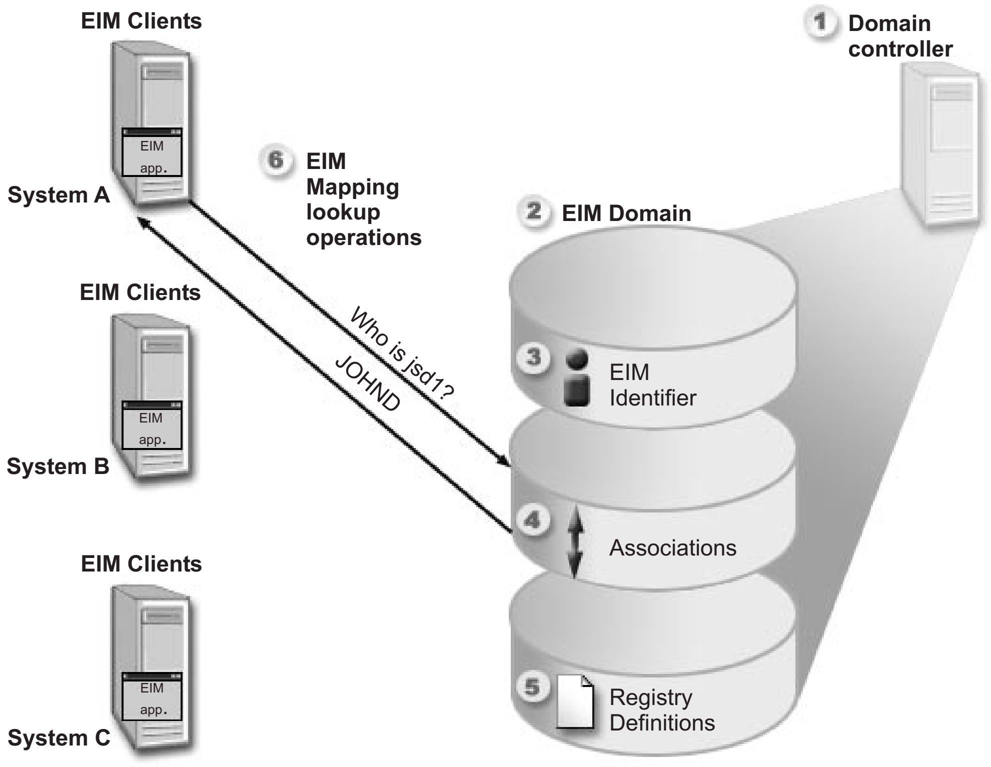


Figure 1. Overview of an EIM implementation. This shows a typical EIM implementation.

Review the following information to learn more about these EIM concepts:
- “EIM domain controller” on page 6
- “EIM domain” on page 6
- “EIM identifier” on page 8
- “EIM registry definition” on page 10
- “EIM associations” on page 13
- “EIM lookup operation” on page 16
- “Authorities” on page 17

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. 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 16 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.

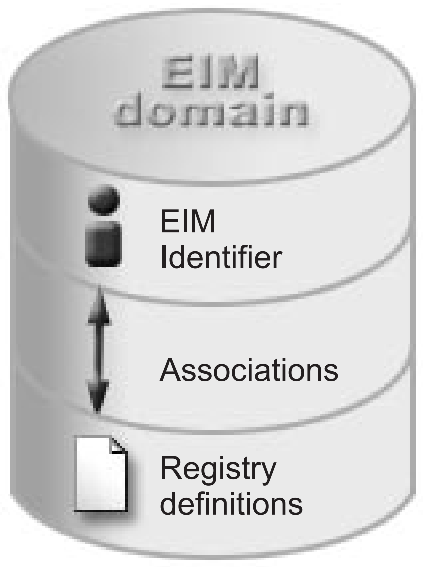


Figure 2. EIM domain and the data that is stored within the domain

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 8 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. See “EIM registry definition” on page 10 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 create associations for identities in user registries that are participating in the EIM domain. Associations provide the information that ties an EIM identifier to a specific user identity in a specific user registry. Consequently, associations must be defined 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. See “EIM lookup operation” on page 16 for more information about this concept.

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`.

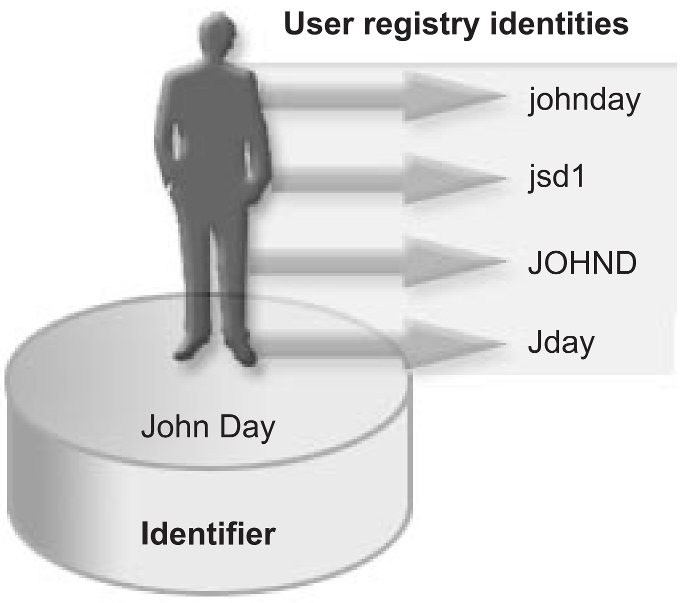


Figure 3. The relationship between the EIM identifier for a real person, John Day, and his various user identities.

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`

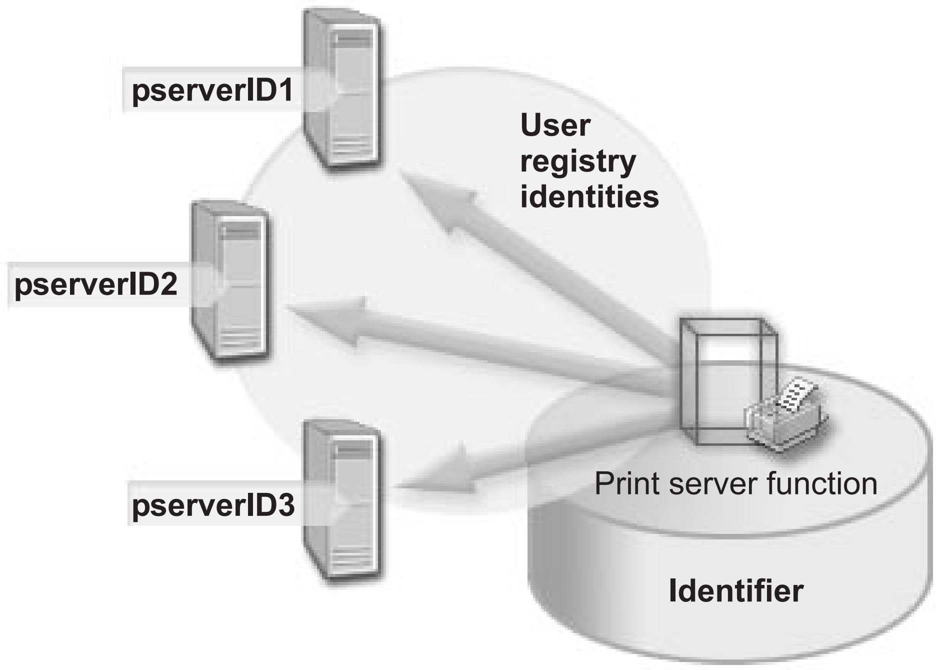


Figure 4. The relationship between the EIM identifier that represents the printer server function, and the various identities for that function.

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.

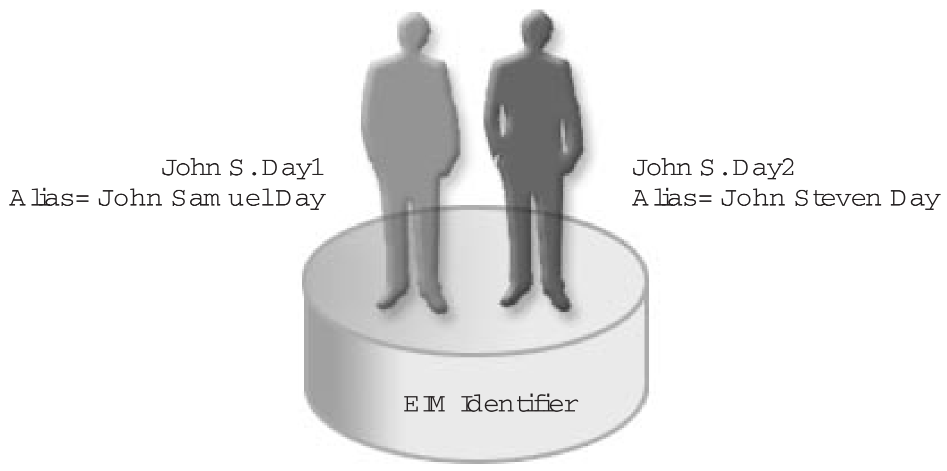


Figure 5. Aliases for the two EIM identifiers based on the shared proper name, John S. Day.

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.

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.

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.

In the following figure, the administrator creates EIM registry definitions for user registries representing System A, System B, and System C. For example, System A contains a user registry for WebSphere Lightweight Third-Party Authentication (LTPA). 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. In addition to the name, the administrator also provides the type of registry as `WebSphere LTPA`.

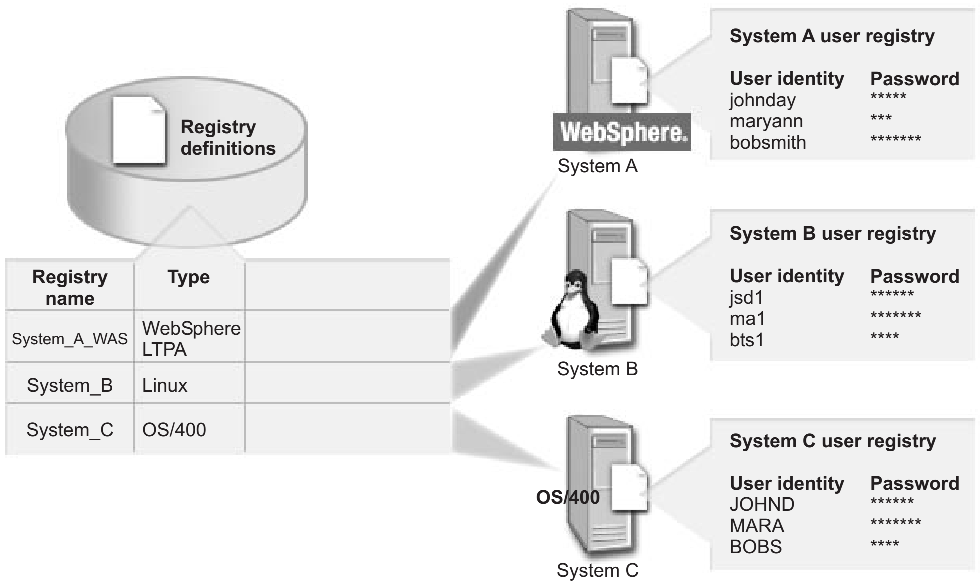


Figure 6. EIM registry definitions for three real-world user registries

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 12.

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 distinguish name (DN)
- Root distinguished name (DN)
- TCP/IP address
- LDAP DNS host name

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 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 "EIM lookup operation" on page 16.

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:

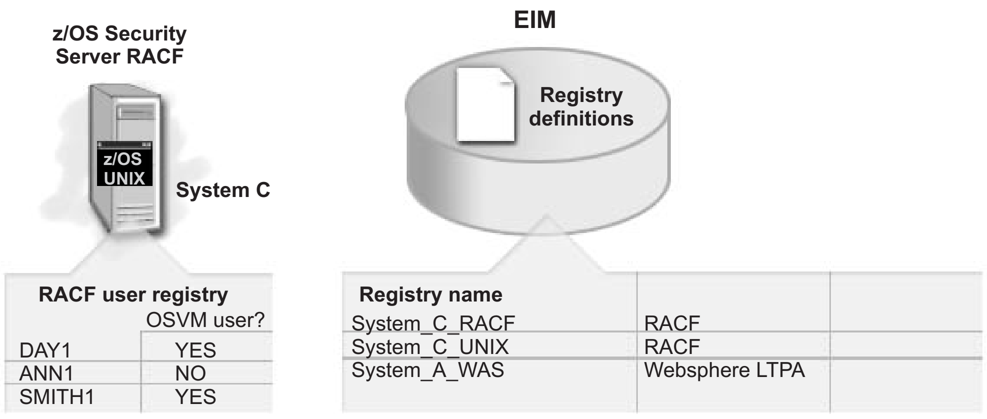
- The registry is provided by an operating system, such as AIX, OS/400, or a security management product such as z/OS Security Server Resource Access Control Facility (RACF).
- The registry contains user identities that are unique to a specific application, such as Lotus Notes.
- The registry contains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

An application registry definition represents 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. 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.



RACF user registry	OSVM user?
DAY1	YES
ANN1	NO
SMITH1	YES

Registry name		
System_C_RACF	RACF	
System_C_UNIX	RACF	
System_A_WAS	Websphere LTPA	

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 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.

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

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.

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 user identities in a single user registry for both authentication and authorization. By using EIM lookup operations, applications now can use user identities in one user registry for authentication while using associated user identities in a different user registry for authorization. Refer to “EIM lookup operation” on page 16 for more information.

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.

target administration
: 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.

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.

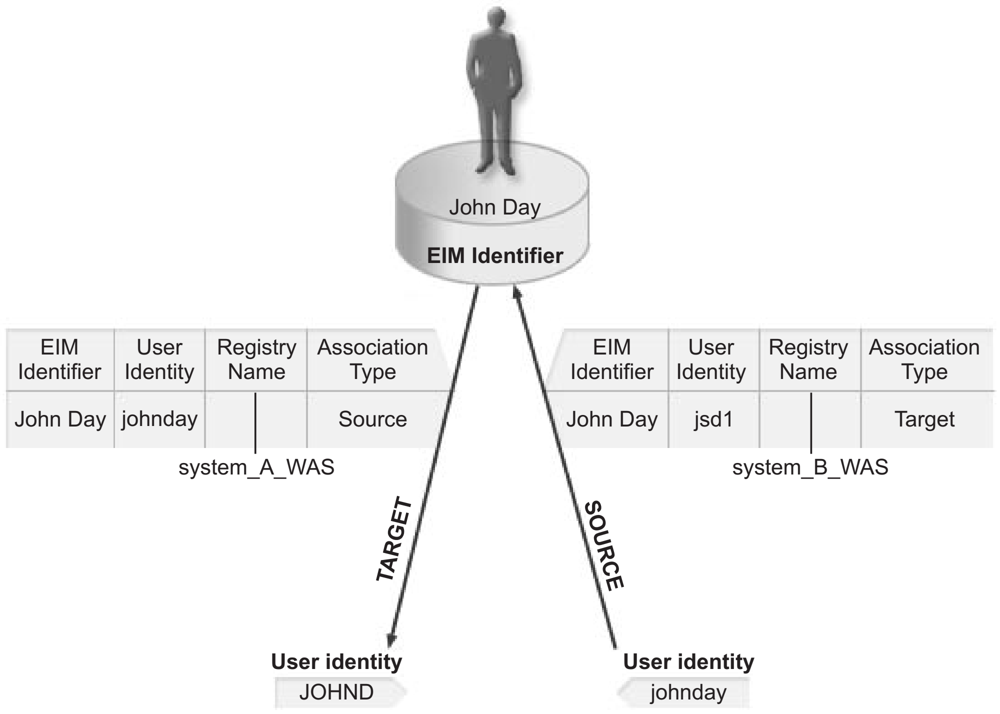


Figure 8. EIM target and source associations for the EIM identifier John Day

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.

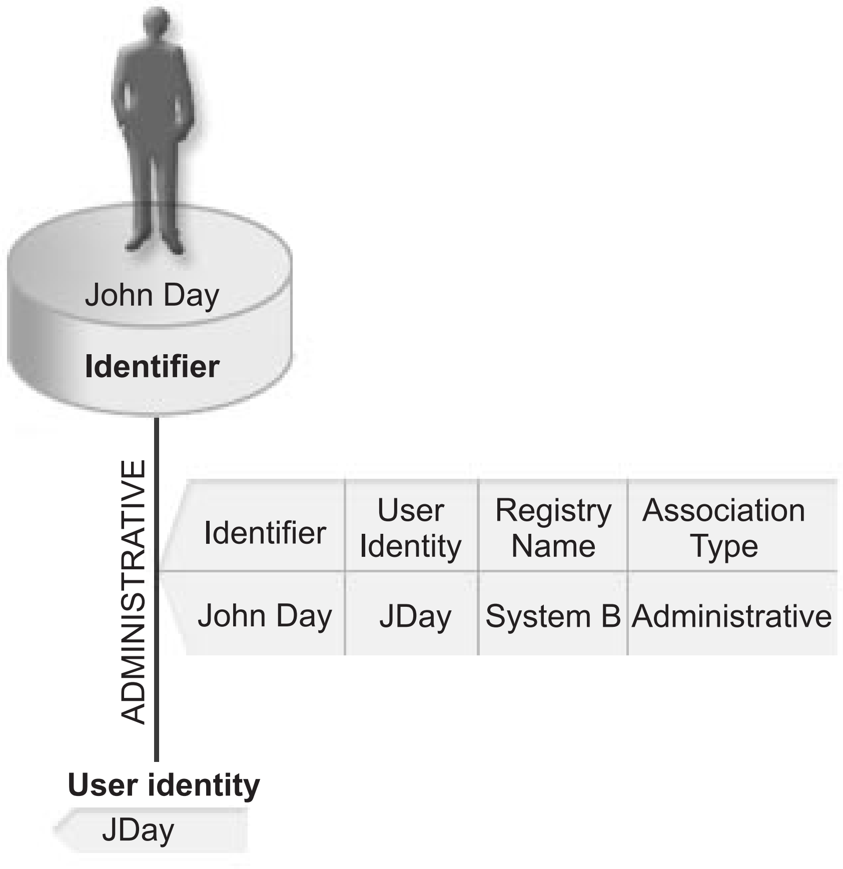


Figure 9. EIM administrative associations for the EIM identifier, John Day

EIM lookup operation

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 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 13 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.

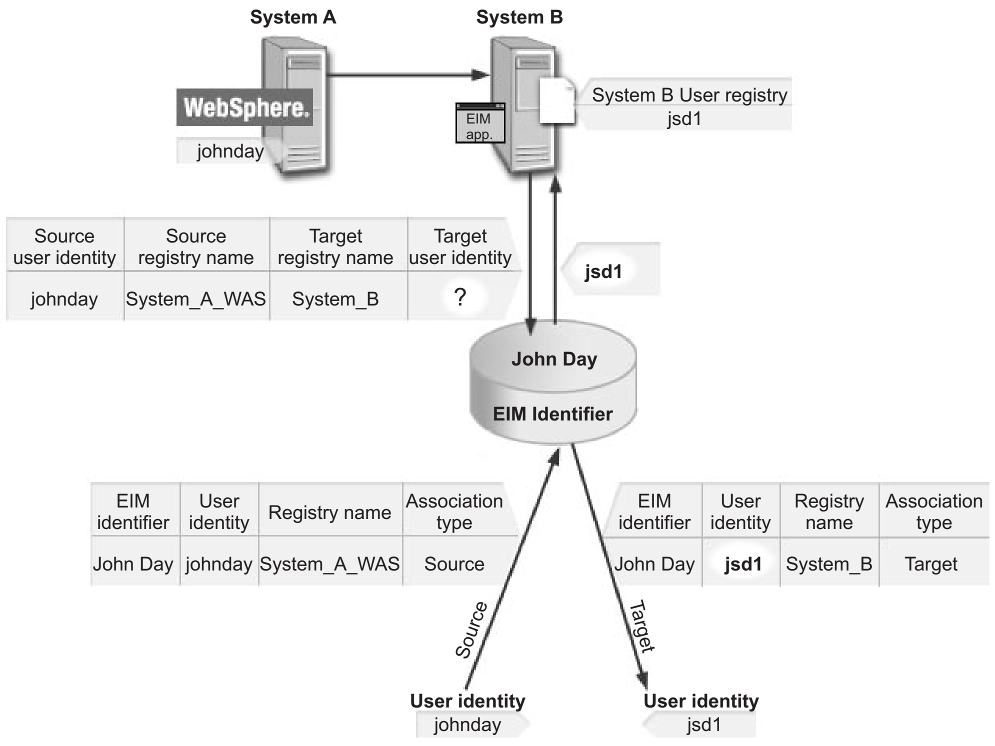


Figure 10. EIM lookup operation based on the known user identity johnday

Authorities

EIM authorities allow a user to perform specific administrative tasks or EIM lookup operations. Only users with EIM administrator authority are allowed to grant or revoke authorities for other users. EIM authorities 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 authority group can perform:

Lightweight Directory Access Protocol (LDAP) administrator

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

- Create a domain
- Delete a domain
- Create and remove EIM identifiers
- Create and remove an EIM registry definition
- 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 authority allows the user to manage all of the EIM data within this EIM domain. A user with this authority can perform the following functions:

- Delete a domain
- Create and remove EIM identifiers
- Create and remove an EIM registry definition
- 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 authority allows the user to add and change EIM identifiers and manage source and administrative associations. A user with this authority 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 authority allows the user to conduct EIM lookup operations. A user with this authority can perform the following functions:

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

EIm registries administrator

This authority allows the user to manage all EIM registry definitions. A user with this authority 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 authority allows the user to manage a specific EIM registry definition. This authority allows a user to:

- Add and remove target associations for the EIM registry definition
- Perform EIM lookup operations

- Retrieve associations, EIM identifiers, and EIM registry definitions

Each of the following tables is organized by the EIM task that the API performs. Each table displays each EIM API, the different EIM authorities, and the access each of these authorities has to certain EIM functions.

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			
eimListIdentifier	X	X	X	X	X	X
eimRemoveIdentifier	X	X				

Table 3. Working with registries

EIM API	LDAP admin	EIM admin	Identifier admin	Identity mapping operations	Registry admin	Admin for selected registries
eimApplicationRegistry	X	X				
eimAddSystemRegistry	X	X				
eimChangeRegistry	X	X			X	X
eimChangeRegistryUser	X	X			X	X
eimChangeRegistryAlias	X	X			X	X
eimListRegistries	X	X	X	X	X	X
eimListRegistryAliases	X	X	X	X	X	X
eimListRegistyUsers	X	X	X	X	X	X
eimRemoveRegisty	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			
eimAddAssociation (source)	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 (target)	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 Ientifier	X	X	X	X	X	X
eimGetRegistryFromAlias	X	X	X	X	X	X
eimGetTargetFromIdentifier	X	X	X	X	X	X
eimGetTargetFromSource	X	X	X	X	X	X

Table 6. 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. 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 7. 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 7. 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 8. EIM APIs software and hardware prerequisites

EIM APIs	LDAP client	Platform
Included in AIX	AIX V5R2	AIX
z/OS Integrated Security Services Enterprise Identity Mapping (EIM) included in z/OS	z/OS Integrated Security Services LDAP	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: `http://wcs.haw.ibm.com/servers/eserver/security/`	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 9.

Table 9. 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 10. Registry worksheet to help with planning considerations for EIM registries and associations

Registry name	Registry type	Registry alias	Association types required	Registry description

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.

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.

Use the worksheet below as a guide to figuring out the kind of information the EIM administrator will need know about the identifiers for the users of the application. The identifier description and additional information fields are free-form and can be used to for descriptive information about the user.

Table 11. Identifier worksheet to help with planning considerations for identifiers

Unique name	Identifier alias	Identifier description	Additional information

Planning considerations for associations

Associations are defined relationships between an identifier and a user ID in a registry.

Tip: 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.

Tip: 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.

Tip: As you plan for your EIM applications, use the following worksheet 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 12. Registry user worksheet to help with planning considerations for EIM associations

Registry user name	Registry name	Unique identifier name	Association type required	Additional information (target associations only)

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 LDAP server for authentication. The bind distinguished name and bind password are defined by the LDAP 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 LDAP 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 LDAP client and LDAP 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 "Authorities" on page 17 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 21.

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

Bind identity	Bind mechanism	EIM authorities required

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 14 on page 28 lists the LDAP servers that can be used as an EIM domain controller.

Table 14. 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™
z/OS V1R4 Security Server LDAP	z/OS V1R4	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 15. 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`	

| **Rule:** 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 26 for more information.

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 23.

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 16 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 16. 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 31.
Install and configure the EIM administration utility	Refer to “Installing and configuring EIM on z/OS” on page 33.
Use RACF commands to set up and tailor EIM	Refer to Chapter 5, “Using RACF commands to set up and tailor EIM,” on page 47.
Ongoing administration tasks	Refer to “Steps for using the eimadmin utility to manage an EIM domain” on page 34.

Chapter 4. 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:

- “Domain authentication methods” on page 38
- “Steps for installing and configuring the EIM domain controller on z/OS”
- “Installing and configuring EIM on z/OS” on page 33
- “Steps for using the eimadmin utility to manage an EIM domain” on page 34
- “Installation considerations for applications” on page 40

It also explores topics about ongoing administration, such as:

- “Managing registries” on page 41
  - “Adding a system and application registry” on page 41
  - “Removing a registry” on page 41
- “Working with registry aliases” on page 42
  - “Assigning an alias” on page 42
  - “Removing an alias” on page 42
  - “Assigning an alias name to a different registry” on page 43
- “Adding a new user” on page 43
  - “Adding an identifier” on page 43
  - “Adding associations” on page 44
- “Removing a user” on page 45
  - “Removing associations” on page 45
  - “Removing an identifier” on page 45
- “Changing access authority” on page 45
  - “Adding access authorities” on page 45
  - “Removing access authorities” on page 46

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.
2. You will need to refer to *z/OS Integrated Security Services LDAP Server Administration and Use*.

Rule: Also, for the z/OS Integrated Security Services 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.

1. First, perform the following steps to install and configure LDAP:
   a. Use the following table to decide what you first need to do:

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: 1. `schema.user.ldif` 2. `schema.IBM.ldif`
You have LDAP installed and configured for the RDBM backend but not the TDBM backend...	Migrate to the TDBM backend. See *z/OS Integrated Security Services LDAP Server Administration and Use* for details about how to do this.	If you are using RDBM, you must migrate to TDBM. Refer to *z/OS Integrated Security Services LDAP Server Administration and Use*.
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.	

Perform the following steps for the decision you have made.

b. The z/OS Integrated Security Services LDAP server must be configured to accept the different types of bind requests. The information from worksheet Table 13 on page 27 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* for more details.

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

d. Load the schema definitions.

Rule: If you are migrating from a pre-z/OS Version 1 Release 4 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* 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.) The LDAP server on z/OS must be configured with the TDBM 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 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 11 represents a basic z/OS configuration.

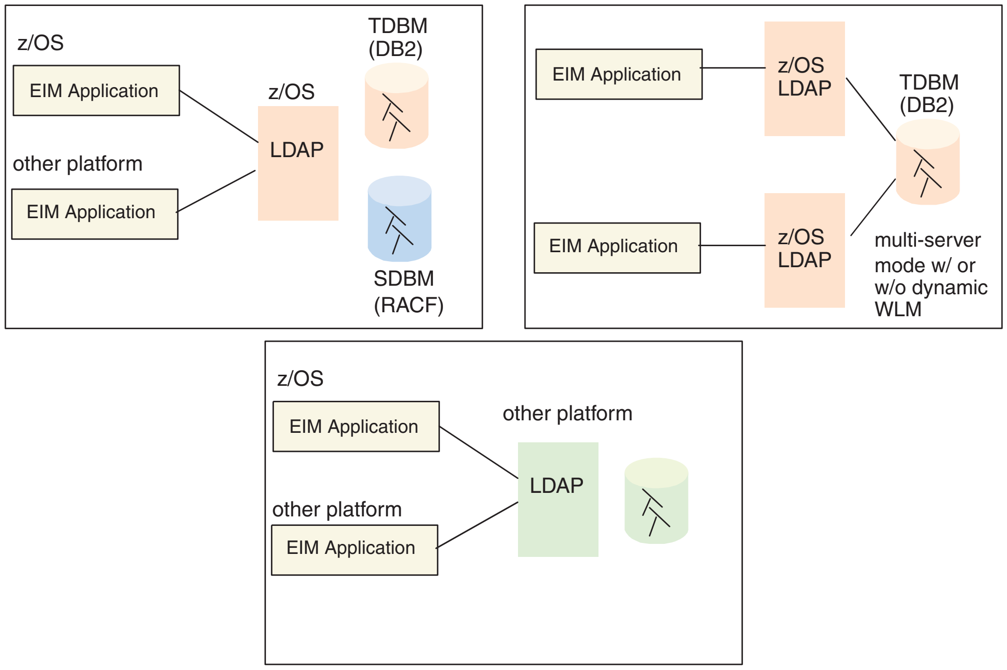


Figure 11. EIM configurations involving z/OS

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 17 on page 34 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 17. HFS install directories

Directory and description	Default value or customized value
Main install directory	`/usr/lpp/eim`
EIM library directory; contains the runtime library(eim.dll) and the definition side deck file(eim.x) for linking EIM applications **Note:** 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 8, "The eimadmin utility," on page 77 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 23 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 38 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 288 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 288 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 84 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 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:

1. You can create associations only after registries and identifiers are in place.
2. 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 on page 37 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 informaton 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 must be APF-authorized. Requiring APF authorization prevents inadvertent or malicious use of EIM APIs to change information in an EIM domain or to extract unauthorized information. See “Preparing to run an EIM application” on page 60 for more information.

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 132) 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 5. 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 18 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 49
  - "Adding a system default using the IRR.EIM.DEFAULTS profile" on page 49
  - "Adding a system default using the IRR.PROXY.DEFAULTS profile" on page 50

Storing LDAP binding information in a profile

Before you begin:

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

Table 18. 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 19. 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 19. 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 FACILTY 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.PROXY.DEFAULTS profile in the FACILITY 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 20. 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 steps to set up EIM so that you do not need a registry name on every lookup:

1. 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.
2. To configure your options, enter the following RACF command:

   ```
   SETROPTS EIMREGISTRY
   ```

   Note: You could also IPL to get the same effect as issuing SETROPTS EIMREGISTRY, but this is not recommended.

For more information on defining a registry name, refer to “EIM registry definition” on page 10.

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 6. 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”
  - “Defining private user registry types in EIM”
    - “Define a private user registry type in EIM”
- “Building an EIM application” on page 59
  - “Compile considerations” on page 59
  - “Link-edit considerations” on page 60
- “Preparing to run an EIM application” on page 60
- “APIs for retrieving the LDAP URL and binding information” on page 60
- “Determining why a mapping is not returned” on page 61

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:
- “Compile considerations”
- “Link-edit considerations” on page 60

Note: The EIM programming interface is provided in a set of C/C++ functions in the EIM DLL. 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 program that uses the EIM API, use the following guidelines.

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, be sure to specify **-Wc,dll** for C++ files, or **-Wc,dll,sscom** for C files that make calls to the EIM APIs or that include the eim.h file.

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`:
1. EBADDATA = Data is not valid
2. EUNKNOWN = Unknown system state
3. ENOTSUP = Operation not supported
4. EBADNAME = The object name specified is not correct
5. ENOTSAFE = The function is not allowed

Refer to “eimErr2String” on page 166 for more information.

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`.

Tip: When linking an EIM application, be sure to specify **-Wl,AC=1** for programs that make calls to the EIM APIs.

Preparing to run an EIM application

Rule: z/OS requires programs that use the EIM APIs to be APF-authorized, which means that you must set the APF-authorization extended attribute for each EIM application program. This attribute is set by using the **extattr** command.

For example **extattr +a eimprog** would set 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*.

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.

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

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 7. 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:

```
ITYnnnn 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 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)

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.

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 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.

ITY0007 Insufficient authority for the operation.

Symbolic Identifier (value): EIMERR_AUTH_ERR (7)

Explanation: Not returned on EIM for z/OS.

Programmer Response: None.

System Action: None.

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 and that a SETROPTS EIMREGISTRY has been issued to bring the registry name into storage. 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.

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.

ITY6001 Application is not APF-authorized.

Symbolic Identifier (value):
EIMERR_ZOS_NO_APF_AUTH (6001)

Explanation: An application that was not APF-authorized called an EIM API that requires APF authorization.

Programmer Response: Ensure the application is linked with AC=1, has the APF extended attribute set and try the service again.

System Action: The called function fails.

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.

Chapter 8. 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 and registry users. 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 89 for more information.) The administrator directs utility processing by specifying a combination of command-line options. (See “Purpose” on page 78 and “Parameters” on page 80 for more information.)

eimadmin

Purpose

Perform actions on the following objects:

- Domains
- Registries
- Identifiers
- Associations
- Access authorities

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
              [-s switch]
              [-v verboseLevel]
              [-c accessType]
              [-f accessUserType]
              [-g registryParent]
              [-i identifier]
              [-j otherIdentifier]
              [-k URI]
              [-n description]
              [-o information]
              [-q accessUser]
              [-r registryName]
              [-t associationType]
              [-u registryUser]
              [-x registryAlias]
              [-y registryType]
              [-z registryAliasType]
              [-d domainDN]
              [-h ldapHost]
              [-b bindDN]
              [-w bindPassword]
              [-K keyFile [ -P keyFilePassword] [-N certificateLabel]]
              [-S connectType]
```

Table 21 on page 79 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
    ```

Notes:

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 89 for more information. See Table 25 on page 91 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 21.

Table 21. Required and optional flags

Object	Action	Required	Optional	Comments
D	a	d, h	n	Add a domain.
	p	d, h	s	Remove a domain. If the domain is not empty, include **'-s RMDEPS'**.
	l	d, h		List domain(s). Specify **-d'*'** to list all domains.
	m	d, h	n	Modify or add a domain attribute.
	e	d, h	n	Remove or clear a domain attribute.
R	a	r,y	g, k, n, x, z	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	Modify or add a registry attribute, including a registry alias.
	e	r	k, n, x, z	Remove or clear a registry attribute, including a registry alias.

Table 21. 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 **'*'**.
		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, 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, 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	n, o	Modify or add an association attribute. Flags **'-n'** and **'-o'** are relevant only to TARGET associations.
	e	r, u	n, o	Remove or clear an association attribute. Flags **'-n'** and **'-o'** are relevant only to TARGET associations.
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

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, stored within an LDAP directory. For more information about EIM domains, see page “EIM domain” on page 6.)

-R A registry. This is the name of a user registry. Associations are defined between identifiers and user IDs in the user registry. (For more information, see page “EIM domain” on page 6.)

-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 6.)

-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 6.)

-C An access authority. This is an EIM-defined LDAP access control group. (For more information, see page “Authorities” on page 17.)

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.

-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.

-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.

-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 83 for a description of *accessUser*.)

KERBEROS
The *accessUser* is a Kerberos identity. (See page 83 for a description of accessUser.)

-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.

-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. 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.

Note: You can specify this option multiple times to define multiple relationships.

-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 42 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

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 10 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 48 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 22 shows the required and optional values for each connect and host type when specified with the eimadmin command:

Table 22. 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 “Authorities” on page 17 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 7, “Messages,” specifically page 71 to page 73.

Error codes

The eimadmin utility returns one of the following exit codes upon completion:

Table 23. 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 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
```

- 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
```

- List identifiers by entering a command such as the following:

```
eimadmin -lI -i "J.C.Smith"
```

  This produces output such as the following:

```
   unique identifier: J.C.Smith
    other identifier: J.C.Smith
    other identifier: Joseph
    other identifier: Joe
         description: 004321
         information: D01
         information: 1990-04-11
```

- List target associations by entering a command such as the following:

```
eimadmin -lA -i "J.C.Smith" -t target
```

  This produces output such as the following:

```
    unique identifier: J.C.Smith
             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 90 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 24. 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 25 on page 91.

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 92 for a more complex example.

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 25. Summary of associated labels

Object Types	Associated Labels	Command Line Options	Descriptions
Domain	DN	-d	Domain distinguished name
	DD	-n	Domain description
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
Identifier	IU	-i	Unique identifier
	IN	-j	Non-unique identifier (multi-value)
	ID	-n	Identifier description
	II	-o	Identifier information (multi-value)
Association	IU	-i	Unique identifier
	RN	-r	Registry name
	UN	-u	Registry user name
	UT	-t	Association type (multi-value)
	UD	-n	User description
	UI	-o	User information (multi-value)
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 25) 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 25 on page 91 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 93 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 10, “EIM header file and example,” on page 271 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:

1. 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.
2. There can be only one UN (registry user name), UD (user description), IU (unique identifier), and ID (identifier description).
3. 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:

1. 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 appearence, without recognizing similarities between them.
2. 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.

Chapter 9. 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, two preliminary sections identify the authority to use the APIs and describe the EIM return code parameter, EimRC.

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.

For information on access authories, refer to “Authorities” on page 17. The following APIs do not require the user to have an EIM authority:
- eimSetConfiguration
- eimRetrieveConfiguration
- eimCreateHandle
- eimDestroyHandle
- eimSetAttribute
- eimGetAttribute
- eimConnect
- eimConnectToMaster

EimRC -- EIM return code parameter

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” on page 95.

Related Information

See also the following:
- “eimListAccess” on page 196
- “eimListUserAccess” on page 239
- “eimQueryAccess” on page 246
- “eimRemoveAccess” on page 250

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
The caller of the API must be APF-authorized.

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. **EIMERR_ZOS_NO_APF_AUTH (6001)** Job Step TCB is not APF-authorized
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 the eimConnect API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master. **EIMERR_READ_ONLY (36)** LDAP connection is for read-only. Use eimConnectToMaster 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, 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. See 84 for a list of predefined types and also look at the eim.h sample (see "eim.h" on page 271). Users can define their own registry types. See "EIM registry definition" on page 10 for details.

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 can 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" on page 95.

Related Information

See also the following:

- "eimAddSystemRegistry" on page 115
- "eimChangeRegistry" on page 128
- "eimListRegistries" on page 221
- "eimRemoveRegistry" on page 262

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
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 in question.

Return Value	Meaning	
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 13 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 retistry 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" on page 95.

Related Information

See also the following:

- "eimGetAssociatedIdentifiers" on page 168
- "eimListAssociations" on page 201
- "eimRemoveAssociation" on page 254

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

The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 the eimConnect API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master. **EIMERR_READ_ONLY (36)** LDAP connection is for read-only. Use eimConnectToMaster 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",
                          "maryjo",
                          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” on page 95.

Related Information

See also the following:

- “eimChangeIdentifier” on page 124
- “eimGetAssociatedIdentifiers” on page 168
- “eimListIdentifiers” on page 214
- “eimRemoveIdentifier” on page 259

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
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 the eimConnect API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	LDAP connection is for read-only. 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 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);
                        .
                        .
                        .
```

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_AIX
- EIM_REGTYPE_NDS
- EIM_REGTYPE_LDAP
- EIM_REGTYPE_POLICY_DIRECTOR
- EIM_REGTYPE_WIN2K

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 with 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" on page 95.

Related Information

See also the following:

- "eimAddApplicationRegistry" on page 102
- "eimChangeRegistry" on page 128
- "eimListRegistries" on page 221
- "eimRemoveRegistry" on page 262

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
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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_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 ports 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.

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 include:

EIM_DOMAIN_DESCRIPTION (0)
Changes the description for the EIM domain. Valid `changeType` is EIM_CHG (0).

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. The *attrName* parameter indicates which type is allowed for each attribute. Valid values include:

EIM_CHG (0)
The attribute is set to the new value.

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" on page 95.

Related Information

See also the following:
- "eimCreateDomain" on page 151
- "eimDeleteDomain" on page 159
- "eimListDomains" on page 207

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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_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 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 connection info parameter of the EIM API does not have a NULL value for the creds field in the connection info 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.	
	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 changes the description of the specified EIM domain:

```
#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;


    /* 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 error = %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 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” on page 95.

Related Information

See also the following:
- “eimAddIdentifier” on page 111
- “eimGetAssociatedIdentifiers” on page 168
- “eimListIdentifiers” on page 214
- “eimRemoveIdentifier” on page 259

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
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 the eimConnect API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	LDAP connection is for read-only. Use eimConnectToMaster 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).

attrValue
: (Input) The new value for the attribute. The value can be a NULL string (for example, ″″).

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).

  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” on page 95.

Related Information

See also the following:
- "eimAddApplicationRegistry" on page 102
- "eimAddSystemRegistry" on page 115
- "eimListRegistries" on page 221
- "eimRemoveRegistry" on page 262

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

The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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_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 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:

```
#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 description for this registry */
   rc = eimChangeRegistry(&handle,
                          "MyAppRegistry",
                          EIM_REGISTRY_DESCRIPTION,
                          "New description",
                          EIM_CHG,
                                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:

- RACF
- OS400
- KERBEROS
- AIX
- NDS
- LDAP
- PD (Policy Director)
- WIN2K

To view the eim.h sample, refer to “eim.h” on page 271. Users can define their own registry alias types. See “EIM registry definition” on page 10 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 to add an alias, and EIM_RMV 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" on page 95.

Related Information

See also the following:

- "eimGetRegistryNameFromAlias" on page 179
- "eimListRegistryAliases" on page 228

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
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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.

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_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)	LDAP connection is for 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 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" on page 95.

Related Information

See also the following:
- "eimListRegistryUsers" on page 233

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
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 the eimConnect API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	LDAP connection is for read-only. Use eimConnectToMaster 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.

| *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
| ports 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.

| *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 is used.

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” on page 95.

Related Information

See also the following:

- "eimConnectToMaster" on page 145
- "eimCreateHandle" on page 156
- "eimDestroyHandle" on page 164
- "eimGetAttribute" on page 175
- "eimSetAttribute" on page 267

Authorization

z/OS authorization
The caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 connection info parameter of the EIM API does not have a NULL value for the creds field in the connection info structure.

Return Value	Meaning	
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.
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 replication 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.

| *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 ports 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.

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 is used.

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” on page 95.

Related Information

See also the following:

- “eimConnect” on page 140
- “eimCreateHandle” on page 156
- “eimDestroyHandle” on page 164
- “eimGetAttribute” on page 175
- “eimSetAttribute” on page 267

Authorization

z/OS authorization
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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_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_CRED_MUST_BE_NULL (58)	The connection info parameter of the EIM API does not have a NULL value for the creds field in the connection info structure.
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 the eimConnect 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 ports 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” on page 95.

Related Information

See also the following:

- "eimChangeDomain" on page 119
- "eimDeleteDomain" on page 159
- "eimListDomains" on page 207

Authorization

EIM data
LDAP administrators have the authority to create an EIM domain.

z/OS authorization
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 object name. The following special characters are not allowed in registry 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_CRED_MUST_BE_NULL (58)	The connection info parameter of the EIM API does not have a NULL value for the creds field in the connection info 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.	
	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
Is the name of the host on which the EIM domain controller is running. (The port number is optional.)

dn
Is the distinguished name of the domain with which to work.

Examples:

```
ldap://systemx:389/ibm-eimDomainName=myEimDomain,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" on page 95.

Related Information

See also the following:

- "eimConnect" on page 140

- “eimConnectToMaster” on page 145
- “eimDestroyHandle” on page 164
- “eimGetAttribute” on page 175
- “eimSetAttribute” on page 267

Authorization

z/OS authorization
The caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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.
EMVSSAFEXTRERR	SAF/RACF EXTRACT error.	
	EIMERR_ZOS_USER_XTR (6002)	RACROUTE REQUEST=EXTRACT error retrieving EIM configuration information from the callers’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.

Return Value	Meaning	
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.
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)	EIM environment is not configured. On z/OS, issue RACF commands to correct the configuration error. 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 ports 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” on page 95.

Related Information

See also the following:

- “eimCreateDomain” on page 151
- “eimChangeDomain” on page 119
- “eimListDomains” on page 207

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

The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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_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_CRED_MUST_BE_NULL (58)	The connection info parameter of the EIM API does not have a NULL value for the creds field in the connection info structure.
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.	
	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" on page 95.

Related Information

See also the following:

- "eimConnect" on page 140
- "eimConnectToMaster" on page 145
- "eimCreateHandle" on page 156
- "eimGetAttribute" on page 175
- "eimSetAttribute" on page 267

Authorization

z/OS authorization
The caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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" on page 95.

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);
    }

      ...
```

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 that the caller provides for the *listData* parameter. The minimum required size 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" on page 95.

Related Information

See also the following:
- "eimAddIdentifier" on page 111
- "eimChangeIdentifier" on page 124
- "eimListIdentifiers" on page 214
- "eimRemoveIdentifier" on page 259

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 caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 the eimConnect API and try the request again.
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 instead of using eimSetConfiguration. 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” on page 95.

Related Information

See also the following:

- “eimConnect” on page 140
- “eimConnectToMaster” on page 145
- “eimCreateHandle” on page 156
- “eimDestroyHandle” on page 164
- “eimSetAttribute” on page 267

Authorization

z/OS authorization
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 the eimConnect 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 84.

aliasValue
(Input) The value of the alias to use for this search.

lengthOfListData
(Input) The number of bytes the caller provides for the *listData* parameter. 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" on page 95.

Related Information

See also the following:
- "eimChangeRegistryAlias" on page 132
- "eimListRegistryAliases" on page 228

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
The caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 sevice 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 the caller provides for the *listData* parameter. 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” on page 95.

Related Information

See also the following:
- “eimGetTargetFromSource” on page 190

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 caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)** Job Step TCB is not APF-authorized.
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.
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_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 could not locate the registry name.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use the eimConnect 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.

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 sevice 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 sevice 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 the caller provides for the listData parameter. 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” on page 95.

Related Information

See also the following:
- “eimGetTargetFromIdentifier” on page 184

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 caller of the API must be APF-authorized.

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.	
	EIMERR_ZOS_NO_APF_AUTH(6001)	Job Step TCB is not APF_AUTHORIZED.
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 could not locate the registry name.
ENOTCONN	LDAP connection has not been made.	
	EIMERR_NOT_CONN (31)	Not connected to LDAP. Use the eimConnect 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");

}
```

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 paramater 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. 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” on page 95.

Related Information

See also the following:

- “eimAddAccess” on page 98
- “eimListUserAccess” on page 239
- “eimQueryAccess” on page 246
- “eimRemoveAccess” on page 250

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
The caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 the eimConnect 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.

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. 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” on page 95.

Related Information

See also the following:

- “eimAddAssociation” on page 106
- “eimGetAssociatedIdentifiers” on page 168
- “eimRemoveAssociation” 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 registry *X* administrator
- EIM mapping lookup

The list returned contains only the information that the user has authority to access.

z/OS authorization
The caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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.
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_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 the eimConnect 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 ports 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. Minimum size required is 20 bytes. The API returns the number of bytes available for the entire list and as much data as space has been provided.

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                   */
} 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” on page 95.

Related Information

See also the following:

- “eimChangeDomain” on page 119
- “eimCreateDomain” on page 151

- "eimDeleteDomain" on page 159

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
The caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)** Job Step TCB is not APF-authorized.
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_NODOMAIN (46)	URL has no domain.
	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 connection info parameter of the EIM API does not have a NULL value for the creds field in the connection info 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 * 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;

    /* Get info for specified domain            */
    if (0 != (rc = eimListDomains(ldapURL,
                                  con,
                                  1000,
                                  list,
                                  err)))
    {
  printf("List domain error = %d\n", rc);
  return -1;
    }

    /* Print the results                        */
    printListResults(list);
    return 0;
}

void printListResults(EimList * list)
{
    int i;
    EimDomain * entry;
    EimListData * listData;
    char * data;
    int dataLength;

    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));

        /* 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 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. 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" on page 95.

Related Information

See also the following:

- "eimAddIdentifier" on page 111
- "eimChangeIdentifier" on page 124
- "eimGetAssociatedIdentifiers" on page 168
- "eimRemoveIdentifier" on page 259

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 caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 the eimConnect 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");

}
```

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).

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 for registry user names.

The predefined registry types that EIM provides include the following:

| • `EIM_REGTYPE_RACF`
| • `EIM_REGTYPE_OS400`
| • `EIM_REGTYPE_KERBEROS_EX`
| • `EIM_REGTYPE_KERBEROS_IG`
| • `EIM_REGTYPE_AIX`
| • `EIM_REGTYPE_NDS`
| • `EIM_REGTYPE_LDAP`
| • `EIM_REGTYPE_POLICY_DIRECTOR`
| • `EIM_REGTYPE_WIN2K`

You can also create your own registry type. This parameter can also be NULL; in this case, the API returns all associations.

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. 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      */
} 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” on page 95.

Related Information

See also the following:

- “eimAddApplicationRegistry” on page 102
- “eimAddSystemRegistry” on page 115
- “eimChangeRegistry” on page 128
- “eimRemoveRegistry” on page 262

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

The caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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>


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         * ldapHost =
       "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";

    char           listData[10000];
    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 registries                       */
    if (0 != (rc = eimListRegistries(&handle,
                                     NULL,
                                     NULL,
                                     EIM_ALL_REGISTRIES,
                                     10000,
                                     list,
                                     err))) {
        printf("List registries 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;
    EimRegistry * 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 = (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));

        /* 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 +

        }

    }

}
```

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. 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” on page 95.

Related Information

See also the following:
- “eimChangeRegistryAlias” on page 132
- “eimGetRegistryNameFromAlias” on page 179

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
The caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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");

}
```

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. 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” on page 95.

Related Information

See also the following:

- “eimChangeRegistryUser” on page 136

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 caller of the API must be APF-authorized.

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.	
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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.

Return Value	Meaning	
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 the eimConnect 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 this 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`. (To connect to an EIM domain using Kerberos information, you need to do so from a non-z/OS platform.)

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. 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" on page 95.

Related Information

See also the following:

- "eimAddAccess" on page 98
- "eimListAccess" on page 196
- "eimRemoveAccess" on page 250
- "eimQueryAccess" on page 246

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
The caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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_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 the eimConnect 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`. (To connect to an EIM domain using Kerberos information, you need to do so from a non-z/OS platform.)

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” on page 95.

Related Information

See also the following:
- “eimAddAccess” on page 98
- “eimListAccess” on page 196
- “eimListUserAccess” on page 239
- “eimRemoveAccess” on page 250

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
The caller of the API must be APF-authorized.

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_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 the eimConnect 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`. (To connect to an EIM domain using Kerberos information, you need to do so from a non-z/OS platform.)

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” on page 95.

Related Information

See also the following:
- “eimAddAccess” on page 98
- “eimListAccess” on page 196
- “eimListUserAccess” on page 239
- “eimQueryAccess” on page 246

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
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 the eimConnect API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	LDAP connection is for read-only. 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 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" on page 95.

Related Information

See also the following:

- "eimAddAssociation" on page 106
- "eimGetAssociatedIdentifiers" on page 168
- "eimListAssociations" on page 201

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
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 the eimConnect API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	LDAP connection is for read-only. 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 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" on page 95.

Related Information

See also the following:

- "eimAddIdentifier" on page 111
- "eimChangeIdentifier" on page 124
- "eimGetAssociatedIdentifiers" on page 168

- "eimListIdentifiers" on page 214

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
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 the eimConnect API and try the request again.
EROFS	LDAP connection is for read-only. Need to connect to master.	
	EIMERR_READ_ONLY (36)	LDAP connection is for read-only. Use eimConnectToMaster 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);
 .
 .
 .
```

eimRemoveRegistry

Purpose

Removes a currently participating registry from the EIM domain.

Note: You cannot remove a system registry if there are any application registries that are a subset of the system registry.

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” on page 95.

Related Information

See also the following:
- “eimAddApplicationRegistry” on page 102
- “eimAddSystemRegistry” on page 115
- “eimChangeRegistry” on page 128
- “eimListRegistries” on page 221

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
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
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 an application registry has this system registry defined.
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.

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 reads the configuration information for the system to use. On z/OS use RACF services instead to retrieve configuration information.

This API retrieves the EIM configuration information for this system.

Format

```
#include <eim.h>


int eimRetrieveConfiguration(unsigned int     lengthOfEimConfig,
                             EimConfig      * configData,
                             int              ccsid,
                             EimRC          * eimrc)
```

Parameters

lengthOfEimConfig
(Input) The number of bytes the caller provides for the configuration information. The minimal 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              */
} EimConfig;
```

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;
```

ccsid
(Input) The coded character set identifier (CCSID) for the output 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" on page 95.

Related Information

See also the following:

- "eimSetConfiguration" on page 269

Authorization

This API requires APF authorization to use.

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. **EIMERR_ZOS_NO_APF_AUTH (6001)** Job Step TCB is not APF-authorized.
ENOTSUP	Operation is not supported. **EIMERR_API_NOTSUPP (6012)** The EIM API is not supported

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" on page 95.

Related Information

See also the following:
- "eimConnect" on page 140
- "eimConnectToMaster" on page 145
- "eimCreateHandle" on page 156
- "eimDestroyHandle" on page 164
- "eimGetAttribute" on page 175

Authorization

z/OS authorization
The caller of the API must be APF-authorized.

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.
	EIMERR_ZOS_NO_APF_AUTH (6001)	Job Step TCB is not APF-authorized.
ENOTSUP	Attribute type is not supported.	
	EIMERR_ATTR_NOTSUPP (6)	The specified attribute is not supported.

eimSetConfiguration

Purpose

This API sets the configuration information for the system to use. On z/OS, use RACF services to set the configuration information. This API has no effect on the configuration.

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" on page 95.

Related Information

None.

Authorization

This API requires APF authorization to use.

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. **EIMERR_ZOS_NO_APF_AUTH (6001)** Job Step TCB is not APF-authorized.
ENOTSUP	Operation is not supported. **EIMERR_API_NOTSUPP (6012)** The EIM API not supported.

Chapter 10. 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 1.7.1.1
 * Last Updated: 4/11/03 10:52:01
 */
/*********************************************************************/
/*                                                                   */
/*    Licensed Materials - Property of IBM                           */
/*    5694-A01                                                       */
/*    (C) Copyright IBM Corp. 2002                                   */
/*    Status = HIT7708                                               */
/*                                                                   */
/*********************************************************************/
#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   */
/*** 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_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_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_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_WIN2K    "1.3.18.0.2.33.9-caseIgnore"

/*--------------------------------------------------------------------*/
/*    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"


/*--------------------------------------------------------------------*/
/*    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 or remove                             */
/*--------------------------------------------------------------------*/
enum EimChangeType {
    EIM_CHG,
    EIM_ADD,
    EIM_RMV
};

enum EimDomainAttr
{                                   /* Change type:                  */
    EIM_DOMAIN_DESCRIPTION          /*     Change                    */
};


enum EimRegistryAttr
```

```
{                                  /* Change type:                */
    EIM_REGISTRY_DESCRIPTION,      /*     Change                  */
    EIM_REGISTRY_LABELEDURI        /*     Change                  */
};

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,                /* Source + target + admin         */
    EIM_TARGET,
    EIM_SOURCE,
    EIM_SOURCE_AND_TARGET,
    EIM_ADMIN
};

/*-------------------------------------------------------------------*/
/*    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;

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
};

typedef struct EimIdentifierInfo
{
    union {
        char       * uniqueName;
        char       * entryUUID;
        char       * name;
    } id;
    enum EimIdType        idtype;
} EimIdentifierInfo;
/*--------------------------------------------------------------------*/
/*                                                                    */
/*    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              */
} 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                    */
} 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       */
} 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            */
} 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                     */
} 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;

/*------------------------------------------------------------------*/
/*  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 eimRetrieveConfiguration
   (
    unsigned int     lengthOfEimConfig, /* Input: size provided for
                                      configData                     */
    EimConfig     * configData,    /* Output: Configuration data
                                      returned.                      */
    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      */
    );



/*-------------------------------------------------------------------*/
/*                                                                   */
/*   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      */
    );

/*-------------------------------------------------------------------*/
/*                                                                   */
/*    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      */
    );



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

Notices

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

Bibliography

The following lists titles and numbers of documents referenced in this publication.

- *z/OS C/C++ Run-Time Library Reference*, SA22-7821
- *z/OS Integrated Security Services LDAP Client Programming*, SC24-5924
- *z/OS Integrated Security Services LDAP Server Administration and Use*, SC24-5923
- *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 System Secure Sockets Layer Programming*, SC24-5901

Index

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

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

Publication No. SA22-7875-01

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
SA22-7875-01

Cut or Fold Along Line

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

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

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

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

SA22-7875-01

Cut or Fold Along Line

Program Number: 5694-A01, 5655-G52

Printed in USA

SA22-7875-01