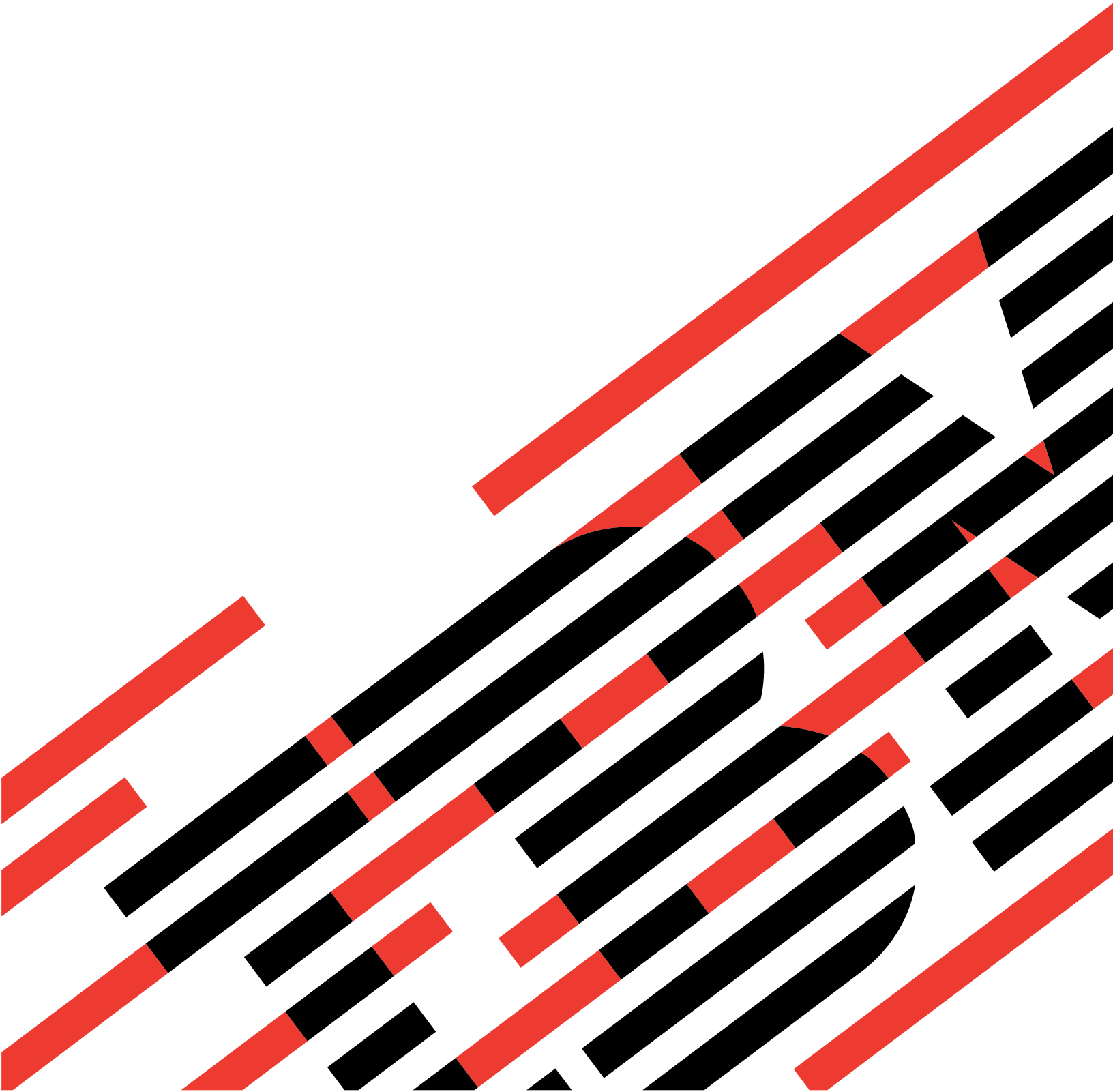
IBM Systems - iSeries

Security -- Enterprise Identity Mapping (EIM) APIs

Version 5 Release 4



IBM Systems - iSeries

Security -- Enterprise Identity Mapping (EIM) APIs

Version 5 Release 4

Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 257.

Sixth Edition (February 2006)

This edition applies to version 5, release 4, modification 0 of IBM i5/OS (product number 5722-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

Contents

Enterprise Identity Mapping (EIM) APIs

Enterprise Identity Mapping (EIM) provides the mechanics for cross-platform single sign-on enablement. Applications can use EIM to perform identity mapping lookup operations to authenticate the user to multiple systems in the enterprise.

For more information on this topic, see Enterprise Identity Mapping.

For information on the EIM return code structure, see "EimRC—EIM Return Code" on page 254.

The Enterprise Identity Mapping APIs are:

- "eimAddGroupRegistry()—Add a Group Registry to the EIM domain" on page 19 (eimAddGroupRegistry()) adds a group registry to the EIM domain.
- "eimAddSystemRegistry()—Add a System Registry to the EIM domain" on page 3 (eimAddSystemRegistry()) adds a system registry to the EIM domain.
- "eimAddApplicationRegistry()—Add an Application Registry to the EIM Domain" on page 7 (eimAddApplicationRegistry()) adds an application registry to the EIM domain.
- "eimAddAccess()—Add EIM Access" on page 11 (eimAddAccess()) adds the user to the EIM access group identified by the access type.
- "eimAddAssociation()—Add EIM Association" on page 15 (eimAddAssociation()) associates a local identity in a specified user registry with an EIM identifier.
- "eimAddIdentifier()—Add EIM Identifier" on page 23 (eimAddIdentifier()) creates an identifier in EIM related to a specific person or entity within an enterprise.
- "eimAddPolicyAssociation()—Add EIM Policy Association" on page 26 (eimAddPolicyAssociation()) adds the specified policy association to the domain.
- "eimAddPolicyFilter()—Add EIM Policy Filter" on page 31 (eimAddPolicyFilter()) adds a policy filter value to the domain.
- "eimChangeDomain()—Change an EIM Domain Object" on page 36 (eimChangeDomain()) changes an attribute for the EIM domain entry identified by domainName.
- "eimChangeIdentifier()— Change EIM Identifier" on page 41 (eimChangeIdentifier()) modifies an existing EIM identifier.
- "eimChangeRegistry()—Change EIM Registry" on page 44 (eimChangeRegistry()) changes the attribute of a registry participating in the EIM domain.
- "eimChangeRegistryAlias()—Change EIM Registry Alias" on page 49 (eimChangeRegistryAlias()) allows you to add or remove a registry alias for the defined registry.
- "eimChangeRegistryUser() —Change EIM Registry User" on page 52 (eimChangeRegistryUser()) changes the attributes of a registry user entry.
- "eimConnect()—Connect to EIM Domain" on page 57 (eimConnect()) is used to connect to the EIM domain that is configured for this platform.
- "eimConnectToMaster()—Connect to EIM Master Domain" on page 60 (eimConnectToMaster()) is used to connect to the EIM master domain controller.
- "eimErr2String()—Convert EimRC into an Error Message" on page 64 (eimErr2String()) converts the EIM return code structure returned by an EIM function into a NULL-terminated error message string.
- "eimCreateDomain()—Create an EIM Domain Object" on page 65 (eimCreateDomain()) creates an EIM domain object on the specified EIM domain controller.
- "eimCreateHandle()—Create an EIM Handle" on page 70 (eimCreateHandle()) is used to allocate an EimHandle structure, which is used to identify the EIM connection and to maintain per-connection information.

- "eimDeleteDomain()—Delete an EIM Domain Object" on page 72 (eimDeleteDomain()) deletes the EIM domain information.
- "eimDestroyHandle()—Destroy an EIM Handle" on page 76 (eimDestroyHandle()) is used to deallocate an EimHandle structure.
- "eimFormatPolicyFilter()—Format EIM Policy Filter" on page 78 (eimFormatPolicyFilter()) generates a policy filter value.
- "eimFormatUserIdentity()—Format User Identity" on page 86 (eimFormatUserIdentity()) formats user identity information for use with other EIM functions.
- "eimGetAssociatedIdentifiers() —Get Associated EIM identifiers" on page 91 (eimGetAssociatedIdentifiers()) returns a list of the identifiers.
- "eimGetAttribute()—Get EIM attributes" on page 98 (eimGetAttribute()) is used to get attributes for this EIM handle.
- "QsyGetEIMConnectInfo()—Get EIM Connect Information" on page 102 (QsyGetEIMConnectInfo()) returns the connection information that will be used by the i5/OS operating system when it needs to connect to the EIM domain that is configured for this system or for the master system.
- "QsyGetEIMHandle()—Get EIM Handle Connected For System" on page 107 (QsyGetEIMHandle()) allocates an EimHandle structure that is connected to EIM.
- "eimGetRegistryNameFromAlias() —Get EIM Registry Name from an Alias" on page 109 (eimGetRegistryNameFromAlias()) returns a list of registry names that match the search criteria provided by*aliasType*and*aliasValue*.
- "eimGetTgtCredsFromIdentifier() —Get EIM Target Identities and Credentials from the Identifier" on page 121 (eimGetTgtCredsFromIdentifier()) gets the target identity or identities and credentials for the specified registry that is associated with the specified EIM identifier.
- "eimGetTargetCredsFromSource() —Get EIM Target Identities and Credentials from the Source" on page 114 (eimGetTargetCredsFromSource()) gets the target identity(ies) and credentials associated with the source identity as defined by source registry name and source registry user.
- "eimGetTargetFromIdentifier() —Get EIM Target Identities from the Identifier" on page 128 (eimGetTargetFromIdentifier()) gets the target identity or identities for the specified registry that is associated with the specified EIM identifier.
- "eimGetTargetFromSource() —Get EIM Target Identities from the Source" on page 134 (eimGetTargetFromSource()) gets the target identity or identies associated with the source identity as defined by source registry name and source registry user.
- "eimGetVersion()—Get EIM Version" on page 140 (eimGetVersion()) returns the EIM version.
- "eimListAccess()—List EIM Access" on page 143 (eimListAccess()) lists the users that have the specified EIM access type.
- "eimListAssociations()— List EIM Associations" on page 148 (eimListAssociations()) returns a list of associations for a given EIM identifier.
- "eimListDomains()—List EIM Domain Objects" on page 154 (eimListDomains()) can be used to list information for a single EIM domain or list information for all EIM domains that can be reached from this platform in the network.
- "eimListIdentifiers()— List EIM Identifiers" on page 160 (eimListIdentifiers()) returns a list of identifiers in the EIM domain.
- "eimListPolicyFilters()—List EIM Policy Filters" on page 166 (eimListPolicyFilters()) lists the policy filters for the domain.
- "eimListRegistries()—List EIM Registries" on page 171 (eimListRegistries()) lists the user registries participating in the EIM domain.
- "eimListRegistryAliases()—List EIM Registry Aliases" on page 179 (eimListRegistryAliases()) returns a list of all the aliases defined for a particular registry.
- "eimListRegistryAssociations()—List EIM Registry Associations" on page 184 (eimListRegistryAssociations()) lists association information for the registry or domain.

- "eimListRegistryUsers()— List EIM Registry Users" on page 191 (eimListRegistryUsers()) lists the users in a particular registry that have target associations defined.
- » "eimListRegistryUsersCreds()— List EIM Registry Users Credentials" on page 197 (eimListRegistryUsersCreds()) lists the users in a particular registry that have target associations defined. «
- "eimListUserAccess()—List EIM User Access" on page 204 (eimListUserAccess()) lists the access groups of which this user is a member.
- "eimQueryAccess()—Query EIM Access" on page 210 (eimQueryAccess()) queries to see if the user has the specified access.
- "eimRemoveRegistry()—Remove a Registry from the EIM Domain" on page 214 (eimRemoveRegistry()) removes a currently participating registry from the EIM domain.
- "eimRemoveAccess()—Remove EIM Access" on page 217 (eimRemoveAccess()) removes the user from the EIM access group identified by the access type.
- "eimRemoveAssociation()— Remove EIM Association" on page 220 (eimRemoveAssociation()) removes an association for a local identity in a specified user registry with an EIM identifier.
- "eimRemoveIdentifier()— Remove EIM Identifier" on page 224 (eimRemoveIdentifier()) removes an EIM identifier and all of its associated mappings from the EIM domain.
- "eimRemovePolicyAssociation()—Remove EIM Policy Association" on page 228 (eimRemovePolicyAssociation()) removes the specified policy association form the domain.
- "eimRemovePolicyFilter()—Remove EIM Policy Filter" on page 232 (eimRemovePolicyFilter()) removes a policy filter value from the domain.
- "eimRetrieveConfiguration()—Retrieve EIM Configuration" on page 235 (eimRetrieveConfiguration()) retrieves the EIM configuration information for this system.
- "eimSetAttribute()—Set EIM attributes" on page 239 (eimSetAttribute()) is used to set attributes in the EIM handle structure.
- "eimSetConfiguration()—Set EIM Configuration" on page 241 (eimSetConfiguration()) sets the configuration information for use by the system.
- "eimSetConfigurationExt()—Set EIM Configuration Extended" on page 245 (eimSetConfigurationExt()) sets the configuration information for use by the system.
- "QsySetEIMConnectInfo()—Set EIM Connect Information" on page 250 (QsySetEIMConnectInfo()) defines the connection information that will be used by the i5/OS operating system when it needs to connect to the EIM domain that is configured for this system or for the master system.

Top | Security APIs | APIs by category

APIs

These are the APIs for this category.

eimAddSystemRegistry()—Add a System Registry to the EIM domain

Syntax

```
#include <eim.h>

int eimAddSystemRegistry(EimHandle      * eim,
                         char           * registryName,
                         char           * registryType,
                         char           * description,
                         char           * URI,
    EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimAddSystemRegistry()** function adds a system registry to the EIM domain. Once added, this registry is participating in the EIM domain. Mapping associations can only be made with identities in registries that are currently participating in the EIM domain.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

registryName (Input)

The name for this system registry. This name needs to be unique within the EIM domain.

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 eim.h for a list of predefined types. A user can define their own registry type. Refer to Registry Type section below.

» A type of EIM_REGTYPE_GROUP_REGISTRY_IG (″1.3.18.0.2.33.17-caseIgnore″) or EIM_REGTYPE_GROUP_REGISTRY_EX (″1.3.18.0.2.33.18-caseExact″) cannot be specified. «

description (Input)

The description for this new system registry entry. This parameter may be NULL.

URI (Input)

The ldap URI (Universal Resource Identifier) needed to access local users in this registry by way of ldap. This parameter may be NULL.

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 may be NULL. For the format of the structure, see “EimRC—EIM Return Code” on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES

Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EEXIST
EIM registry entry already exists.

EIMERR_REGISTRY_EXISTS (37) Registry entry already exists in EIM.

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. Please check API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_REGTYPE_INVAL (62)	Registry type is not valid.

ENOMEM
Unable to allocate required space.

EIMERR_NOMEM (27) No memory available. Unable to allocate required space.

ENOTCONN
LDAP connection has not been made.

EIMERR_NOT_CONN (31) Not connected to LDAP. Use 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.

User Defined Registry Type

The registry type is comprised of two pieces: 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. Platforms can define their own registry type. They would first define a unique OID for their registry and then concatenate it with the predefined normalization methods. Refer to eim.h for the supported normalization methods.

Example:

```
#define MYREGOID  "7.6.5.4.3.2.1"
MyRegType = MYREGOID +  EIM_NORM_CASE_IGNORE;
```

Restrictions

There is a restriction on the characters allowed for registry name.

The following characters are special characters that are not allowed in object names. They also should not be used in object attributes that would be used for a search operation.

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

Related Information

- "eimAddApplicationRegistry()—Add an Application Registry to the EIM Domain" on page 7 —Add an Application Registry to the EIM Domain
- "eimAddGroupRegistry()—Add a Group Registry to the EIM domain" on page 19 —Add a Group Registry to the EIM Domain
- "eimRemoveRegistry()—Remove a Registry from the EIM Domain" on page 214 —Remove a Registry from the EIM Domain
- "eimChangeRegistry()—Change EIM Registry" on page 44 —Change EIM Registry
- "eimListRegistries()—List EIM Registries" on page 171 —List EIM Registries

Example

See Code disclaimer information for information pertaining to code examples.

The following example creates a new EIM system registry.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
```

```
    EimRC       * err;
    EimHandle   * handle;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Add new system registry                  */
    if (0 != (rc = eimAddSystemRegistry(handle,
                                        "MyRegistry",
                                        EIM_REGTYPE_OS400,
                                        "The first registry",
                                        NULL,
                                        err)))
        printf("Add system registry error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimAddApplicationRegistry()—Add an Application Registry to the EIM Domain

Syntax

```
#include <eim.h>

int eimAddApplicationRegistry(EimHandle      * eim,
                              char           * registryName,
                              char           * registryType,
                              char           * description,
                              char           * systemRegistryName,
         EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimAddApplicationRegistry()** function adds an application registry to the EIM domain. An application registry is a subset of a system registry. These can be used to manage which applications can be used by a user in a registry. Once added, this registry is participating in the EIM domain. Mapping associations can only be made with identities in registries that are currently participating in the EIM domain.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

eim (Input)
The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

registryName (Input)
The name for this application registry. This name needs to be unique within the EIM domain.

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 eim.h for a list of predefined types. A user can define their own registry type. Refer to Registry Type section below.

» A type of EIM_REGTYPE_GROUP_REGISTRY_IG (″1.3.18.0.2.33.17-caseIgnore″) or EIM_REGTYPE_GROUP_REGISTRY_EX (″1.3.18.0.2.33.18-caseExact″) cannot be specified.

«

description (Input)
The description for this new application registry entry. This parameter may be NULL.

systemRegistryName (Input)
The name of the system registry of which this application registry is a subset.

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 may be NULL. For the format of the structure, see “EimRC—EIM Return Code” on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EEXIST
EIM registry entry already exists.

EIMERR_REGISTRY_EXISTS (37) Registry entry already exists in EIM.

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 for a list of restricted characters.
EIMERR_PARM_REQ (34) Missing required parameter. Please check API documentation.
EIMERR_PTR_INVAL (35) Pointer parameter is not valid.
» *EIMERR_REGTYPE_INVAL (62)* Registry type 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 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.

User Defined Registry Type

The registry type is comprised of two pieces: 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. Platforms can define their own registry type. They would first define a unique OID for their registry and then concatenate it with the predefined normalization methods. Refer to eim.h for the supported normalization methods.

Example:

```
#define MYREGOID  "7.6.5.4.3.2.1"
MyRegType = MYREGOID +  EIM_NORM_CASE_IGNORE;
```

Restrictions

There is a restriction on the characters allowed for registry name.

The following characters are special characters that are not allowed in object names. They also should not be used in object attributes that would be used for a search operation.

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

Related Information

- "eimAddSystemRegistry()—Add a System Registry to the EIM domain" on page 3 —Add a System Registry to the EIM Domain
- "eimAddGroupRegistry()—Add a Group Registry to the EIM domain" on page 19 —Add a Group Registry to the EIM Domain
- "eimRemoveRegistry()—Remove a Registry from the EIM Domain" on page 214 —Remove a Registry from the EIM Domain
- "eimChangeRegistry()—Change EIM Registry" on page 44 —Change EIM Registry
- "eimListRegistries()—List EIM Registries" on page 171 —List EIM Registries

Example

See Code disclaimer information for information pertaining to code examples.

The following example creates a new EIM application registry.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Add new application registry             */
```

```
    if (0 != (rc = eimAddApplicationRegistry(handle,
                                             "MyXXXRegistry",
                                             EIM_REGTYPE_OS400,
                                             "For XXX applications",
                                             "MyRegistry",
                                             err)))
        printf("Add application registry error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimAddAccess()—Add EIM Access

Syntax

```
#include <eim.h>

int eimAddAccess(EimHandle          * eim,
                 EimAccessUser      * accessUser,
                 enum EimAccessType   accessType,
                 char               * registryName,
   EimRC              * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimAddAccess()** function adds the user to the EIM access group identified by the access type.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

accessUser (Input)

A structure that contains the user information for which to add access.

EIM_ACCESS_LOCAL_USER indicates a local user name on the system that the API is run. The local user name will be converted to the appropriate access id for this system.

EIM_ACCESS_KERBEROS indicates a kerberos principal. The kerberos principal will be converted to the appropriate access id. For example, petejones@therealm will be converted to ibm-kn=petejones@therealm.

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.
» EIM_ACCESS_CREDENTIAL_DATA (5)	Authority to retrieve credential data. «

registryName (Input)
The name of the registry for which to add access. This parameter is only used if EimAccessType is EIM_ACCESS_REGISTRY. If EimAccessType is anything other than EIM_ACCESS_REGISTRY, this parameter must be NULL.

eimrc (Input)
The structure in which to return error code information. If the return value is not 0, eimrc is set with additional information. This parameter may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1)	Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26)	Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13)	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 API documentation.
EIMERR_PTR_INVAL (35)	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 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.

Related Information

- "eimRemoveAccess()—Remove EIM Access" on page 217 —Remove EIM Access
- "eimListAccess()—List EIM Access" on page 143 —List EIM Access
- "eimListUserAccess()—List EIM User Access" on page 204 —List EIM User Access
- "eimQueryAccess()—Query EIM Access" on page 210 —Query EIM Access

Example

See Code disclaimer information for information pertaining to code examples.

The following example adds users to access groups.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    EimAccessUser user;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Set up access user information           */
    user.userType = EIM_ACCESS_DN;
    user.user.dn="cn=pete,o=ibm,c=us";

    /* Add access for this user.                */
    if (0 != (rc = eimAddAccess(handle,
                                &user,
                                EIM_ACCESS_ADMIN,
                                NULL,
                                err)))
    {
        printf("Add access error = %d", rc);
        return -1;
    }

    /* Set up access user information           */
    user.userType = EIM_ACCESS_LOCAL_USER;
    user.user.dn="mjjones";

    /* Add access for this user.                */
    if (0 != (rc = eimAddAccess(handle,
                                &user,
                                EIM_ACCESS_REGISTRY,
                                "MyRegistry",
                                err)))
    {
        printf("Add access error = %d", rc);
        return -1;
    }

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

eimAddAssociation()—Add EIM Association

Syntax

```
#include <eim.h>

int eimAddAssociation(EimHandle               * eim,
                      enum EimAssociationType   associationType,
                      EimIdentifierInfo       * idName,
                      char                    * registryName,
                      char                    * registryUserName,
        EimRC                   * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimAddAssociation()** function associates a local identity in a specified user registry with an EIM identifier. EIM supports three kinds of associations: source, target, and administrative. All EIM associations are between an EIM identifier and a local user identity — never directly between local user identities.

Associated source identities are user identities that are primarily for authentication purposes. They can be used as the source identity of a mapping lookup operation (that is, eimGetTargetFromSource()), but will not be found as the target of a mapping lookup operation.

Associated target identities are user identities that are primarily used to secure existing data. They will be found as the result of a mapping lookup operation, but cannot be used as the source identity for a mapping lookup operation.

Administrative associations are used to show that an identity is associated with an EIM identifier, but cannot be used as the source for, and will not be found as the target of, a mapping lookup operation.

A single user identity may be used as both a target and a source. This is done by creating both a source and a target association for the local user identity with the appropriate EIM identifier. While this API supports an association type of EIM_SOURCE_AND_TARGET, two associations are actually created.

For an EIM identifier to be useful in mapping lookup operations, it must have at least one “source” and at least one “target” association.

» See EIM Mapping Lookup Algorithm for the affect that associations have on the mapping lookup operation. «

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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 authority to an individual registry

Parameters

eim (Input)
The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

associationType (Input)
The type of association to be added. 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` indicates which identifier name is provided. Use of the `uniqueName` provides the best performance. Specifying an `idtype` of EIM_NAME does not guarantee that a unique EIM identifier will be found. Therefore, use of EIM_NAME may result in an error.

registryName (Input)
The registry name for the association.

registryUserName (Input)
The registry user name for the association. The registry user name may be normalized according to the normalization method for defined registry.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry or identifier name is not valid or insufficient access to EIM data.

EIMERR_IDNAME_AMBIGUOUS (20) More than 1 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) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35) Pointer parameter is not valid.

ENOMEM
Unable to allocate required space.

EIMERR_NOMEM (27) No memory available. Unable to allocate required space.

ENOTCONN
LDAP connection has not been made.

EIMERR_NOT_CONN (31) Not connected to LDAP. Use 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_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimGetAssociatedIdentifiers() —Get Associated EIM identifiers" on page 91 —Get Associated EIM Identifiers
- "eimRemoveAssociation()— Remove EIM Association" on page 220—Remove an EIM Association
- "eimListAssociations()— List EIM Associations" on page 148—List EIM Associations

Example

See Code disclaimer information for information pertaining to code examples.

The following example creates 3 associations for the same identifier: administrative, source and target.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    EimIdentifierInfo x;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Set up identifier information            */
    x.idtype = EIM_UNIQUE_NAME;
    x.id.uniqueName = "mjones";

    /* Add an admin association                 */
    if (0 != (rc = eimAddAssociation(handle,
                                     EIM_ADMIN,
                                     &x,
                                     "MyRegistry",
                                     "maryjones",
                                     err)))
    {
```

```
        printf("Add Association error = %d", rc);
        return -1;
    }
    /* Add a source association                */
    if (0 != (rc = eimAddAssociation(handle,
                                     EIM_SOURCE,
                                     &x,
                                     "kerberosRegistry",
                                     "mjjones",
                                     err)))
    {
        printf("Add Association error = %d", rc);
        return -1;
    }
    /* Add a target association                */
    if (0 != (rc = eimAddAssociation(handle,
                                     EIM_TARGET,
                                     &x,
                                     "MyRegistry",
                                     "marymjo",
                                     err)))
    {
        printf("Add Association error = %d", rc);
        return -1;
    }

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimAddGroupRegistry()—Add a Group Registry to the EIM domain

Syntax

```
#include <eim.h>

int eimAddGroupRegistry(EimHandle      * eim,
                        char           * registryName,
                        char           * registryType,
                        char           * description,
                        EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimAddGroupRegistry()** function adds a group registry to the EIM domain. Once added, this registry is participating in the EIM domain. Mapping associations can only be made with identities in registries that are currently participating in the EIM domain.

A group registry can be used when a person has the same identity on multiple systems in the enterprise. You can define one association to the group registry that is used for all registries that are members of the group registry. The members of the group registry must have the same case sensitivity attribute (-caseIgnore or -caseExact) as the group. A group registry cannot be a member of another group registry. You can manage the members of a group registry using the Change EIM Registry (eimChangeRegistry) API.

See EIM Mapping Lookup Algorithm to see how associations to group registries are used in the mapping lookup operation.

EIM version 3 must be supported by the local EIM APIs to use this API (see “eimGetVersion()—Get EIM Version” on page 140—Get EIM Version).

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

registryName (Input)

The name for this group registry. This name needs to be unique within the EIM domain.

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 eim.h for a list of predefined types. The type must be either EIM_REGTYPE_GROUP_REGISTRY_IG (″1.3.18.0.2.33.17-caseIgnore″) or EIM_REGTYPE_GROUP_REGISTRY_EX (″1.3.18.0.2.33.18-caseExact″).

description (Input)

The description for this new group registry entry. This parameter may be NULL.

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 may be NULL. For the format of the structure, see “EimRC—EIM Return Code” on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES

Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1)	Insufficient access to EIM data.

EBADDATA

eimrc is not valid.

EBUSY

Unable to allocate internal system object.

EIMERR_NOLOCK (26)	Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EEXIST
EIM registry entry already exists.

EIMERR_REGISTRY_EXISTS (37) Registry entry already exists in EIM.

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. Please check API documentation.
EIMERR_PTR_INVAL (35) Pointer parameter is not valid.
EIMERR_REGTYPE_INVAL (62) Registry type is not valid.
EIMERR_FUNCTION_NOT_SUPPORTED (70) The specified function is not supported by the EIM version.

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 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.

Restrictions

There is a restriction on the characters allowed for a registry name.

The following characters are special characters that are not allowed in object names. They also should not be used in object attributes that would be used for a search operation.

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

Related Information

- "eimAddSystemRegistry()—Add a System Registry to the EIM domain" on page 3 —Add a System Registry to the EIM Domain
- "eimAddApplicationRegistry()—Add an Application Registry to the EIM Domain" on page 7 —Add an Application Registry to the EIM Domain
- "eimRemoveRegistry()—Remove a Registry from the EIM Domain" on page 214 —Remove a Registry from the EIM Domain
- "eimChangeRegistry()—Change EIM Registry" on page 44 —Change EIM Registry
- "eimListRegistries()—List EIM Registries" on page 171 —List EIM Registries

Example

See Code disclaimer information for information pertaining to code examples.

The following example creates a new EIM group registry.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Add new group registry                   */
    if (0 != (rc = eimAddGroupRegistry(handle,
                                       "MyGroupRegistry",
                                       EIM_REGTYPE_GROUP_REGISTRY_IG,
                                       "The group registry",
                                       err)))
        printf("Add group registry error = %d", rc);

    return 0;
}
```

API introduced: V5R4

eimAddIdentifier()—Add EIM Identifier

Syntax

```
#include <eim.h>

int eimAddIdentifier(EimHandle       * eim,
                     char            * name,
                     enum EimIdAction  nameInUseAction,
                     unsigned int    * sizeOfUniqueName,
                     char            * uniqueName,
                     char            * description,
       EimRC           * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimAddIdentifier()** function 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.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

name (Input)

A name to be used for this identifier.

nameInUseAction (Input)

The name for the new identifier must be unique. This value indicates the action to be taken if the provided name is already being used. 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. This parameter is ignored if *nameInUseAction* is EIM_FAIL.

At input it is the size provided by the caller. 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 to return the unique identifier for this new EIM identifier. This parameter is ignored if *nameInUseAction* is EIM_FAIL.

description (Input)
Description for the new EIM identifier. This parameter may be NULL.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EEXIST
Identifier already exists.

EIMERR_IDENTIFIER_EXISTS (19) EIM Identifier already exists by this name.

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 API documentation.
EIMERR_PTR_INVAL (35) 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 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.

Restrictions

There is a restriction on the characters allowed for identifier name.

The following characters are special characters that are not allowed in object names. They also should not be used in object attributes that would be used for a search operation.

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

Related Information

- "eimRemoveIdentifier()— Remove EIM Identifier" on page 224—Remove EIM Identifier
- "eimChangeIdentifier()— Change EIM Identifier" on page 41—Change EIM Identifier
- "eimListIdentifiers()— List EIM Identifiers" on page 160—List EIM Identifiers
- "eimGetAssociatedIdentifiers() —Get Associated EIM identifiers" on page 91 —Get Associated EIM Identifiers

Example

See Code disclaimer information for information pertaining to code examples.

The following example will add an EIM identifier.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
```

```
    EimHandle   * handle;

    char          unique[30];
    unsigned int  sizeOfUnique = 30;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Add new identifier of Mary Smith        */
    if (0 != (rc = eimAddIdentifier(handle,
                                    "Mary Smith",
                                    EIM_GEN_UNIQUE,
                                    &sizeOfUnique,
                                    unique,
                                    "The coolest person",
                                    err)))
        printf("Add identifier error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

eimAddPolicyAssociation()—Add EIM Policy Association

Syntax

```
#include <eim.h>

int eimAddPolicyAssociation(EimHandle                * eim,
                            EimPolicyAssociationInfo  * policyAssoc,
                            EimRC                     * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimAddPolicyAssociation()** function adds the specified policy association to the domain. A policy association is used to specify the target association for a mapping lookup operation without having to define specific source associations for all users. A policy association will be used in a mapping lookup operation (eimGetTargetFromSource or eimGetTargetFromIdentifier) if a specific source association does not exist.

» EIM version 2 must be supported by the local EIM APIs to use this API (see "eimGetVersion()—Get EIM Version" on page 140—Get EIM Version). «

There are 3 types of policy associations that are supported:

1. Certificate filter policy associations
2. Default registry policy associations
3. Default domain policy associations

A certificate filter policy association is used to map user (or client) certificates with similar attributes to the same target identity in the target registry. For example, a certificate filter policy association can be added so that all certificates issued by the same Certificate Authority (CA) are mapped to the same target identity in the target registry. Or, all certificates from the same organization are mapped to the same target identity in the target registry.

A default registry policy association is used to map any user in the specified source registry to the same target identity in the target registry.

A default domain association policy is used to map all users to the same target identity in the target registry.

The use of policy associations is controlled by the version of the API interface, not the domain. If policy associations are added to a domain, they will only be used in a mapping lookup operation if the version of the mapping lookup API that is used to access the domain supports policy associations. » See EIM Mapping Lookup Algorithm for the affect that policy associations have on the mapping lookup operation.

In the mapping lookup algorithm, there is a check to see if there is a certificate policy filter value that matches the source identity. « To locate a certificate policy filter value, a search will be done using a series of full and partial distinguished names (DNs) until the most specific matching certificate policy filter value is found. The following values are used in sequence to search for a matching certificate policy filter value:

1. *<SDN>subject's-full-DN</SDN><IDN>issuer's-full-DN</IDN>*
2. *<SDN>subject's-partial-DN</SDN><IDN>issuer's-full-DN</IDN>*
3. *<SDN>subject's-full-DN</SDN>*
4. *<SDN>subject's-partial-DN</SDN>*
5. *<IDN>issuer's-full-DN</IDN>*
6. *<IDN>issuer's-partial-DN</IDN>*

Note that searching is not done for the following values:

- *<SDN>subject's-full-DN</SDN><IDN>issuer's-partial-DN</IDN>*
- *<SDN>subject's-partial-DN</SDN><IDN>issuer's-partial-DN</IDN>*

Each step of the search using a partial DN may actually involve a series of searches for partial name values based on the full DN. Each partial DN value in the series is determined by removing the next most specific node in the DN. The nodes are removed from the most specific to the least specific, in the order that they appear in the DN.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM authority to an individual registry

  This authority is needed to the target registry.

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

policyAssoc (Input)

The information about the policy association to be added.

The EimPolicyAssociationInfo structure contains information about the policy association to add.

For EIM_CERT_FILTER_POLICY (6) association type, the *policyAssociation* field must contain an EimCertificateFilterPolicyAssociation structure. The *sourceRegistry* field must contain the name of a registry that has a type of X.509. The certificate filter policy value specified in the *filterValue* field must have already been added using the Add EIM Policy Filter (eimAddPolicyFilter) API.

For EIM_DEFAULT_REG_POLICY (7) association type, the *policyAssociation* field must contain an EimDefaultRegistryPolicyAssociation structure.

For EIM_DEFAULT_DOMAIN_POLICY (8) association type. the *policyAssociation* field must contain an EimDefaultDomainPolicyAssociation structure.

The structure layouts follow:

```
enum EimAssociationType {
    EIM_ALL_ASSOC,                 /* Not supported on this interface*/
    EIM_TARGET,                    /* Not supported on this interface*/
    EIM_SOURCE,                    /* Not supported on this interface*/
    EIM_SOURCE_AND_TARGET,         /* Not supported on this interface*/
    EIM_ADMIN,                     /* Not supported on this interface*/
    EIM_ALL_POLICY_ASSOC,          /* Not supported on this interface*/
    EIM_CERT_FILTER_POLICY,        /* Association is a certificate
                                      filter policy association.     */
    EIM_DEFAULT_REG_POLICY,        /* Association is a default
                                      registry policy association    */
    EIM_DEFAULT_DOMAIN_POLICY      /* Policy is a default policy for
                                      the domain.                    */
};

typedef struct EimCertificateFilterPolicyAssociation
{
    char  * sourceRegistry;        /* The source registry to add the
                                      policy association to.         */
    char  * filterValue;           /* The filter value of the policy.*/
    char  * targetRegistry;        /* The name of the target registry
                                      that the filter value should
                                      map to.                        */
    char  * targetRegistryUserName; /* The name of the target registry
                                      user name that the filter value
                                      should map to.                 */
} EimCertificateFilterPolicyAssociation;

typedef struct EimDefaultRegistryPolicyAssociation
{
    char  * sourceRegistry;        /* The source registry to add the
                                      policy association to.         */
    char  * targetRegistry;        /* The name of the target registry
                                      that the policy should map to. */
    char  * targetRegistryUserName; /* The name of the target registry
                                      user name that the policy
                                      should map to.                 */
} EimDefaultRegistryPolicyAssociation;

typedef struct EimDefaultDomainPolicyAssociation
{
    char  * targetRegistry;        /* The name of the target registry
                                      that the policy should map to. */
```

```
    char  * targetRegistryUserName; /* The name of the target registry
                                       user name that the policy
                                       should map to.                */
} EimDefaultDomainPolicyAssociation;

typedef struct EimPolicyAssociationInfo
{
    enum EimAssociationType type;
    union {
        EimCertificateFilterPolicyAssociation   certFilter;
        EimDefaultRegistryPolicyAssociation     defaultRegistry;
        EimDefaultDomainPolicyAssociation       defaultDomain;
    } policyAssociation;
} EimPolicyAssociationInfo;
```

eimrc (Input/Output)
The structure in which to return error code information. If the return value is not 0, eimrc is set with additional information. This parameter may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry name is not valid or insufficient access to EIM data, or policy filter value is not found.

EIMERR_NOREG (28) EIM Registry not found or insufficient access to EIM data.
EIMERR_NOPOLICYFILTER (61) Policy filter value not found for the specified EIM Registry.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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_PARM_REQ (34)	Missing required parameter. Please check API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified function is not supported by the EIM version.

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 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.
EIMERR_UNEXP_OBJ_ VIOLATION (56)	Unexpected object violation.

Related Information

- "eimRemovePolicyAssociation()—Remove EIM Policy Association" on page 228 —Remove EIM Policy Association
- "eimListRegistryAssociations()—List EIM Registry Associations" on page 184 —List EIM Registry Associations
- "eimFormatPolicyFilter()—Format EIM Policy Filter" on page 78 —Format EIM Policy Filter
- "eimAddPolicyFilter()—Add EIM Policy Filter" on page 31 —Add EIM Policy Filter
- "eimChangeDomain()—Change an EIM Domain Object" on page 36 —Change EIM Domain
- "eimChangeRegistry()—Change EIM Registry" on page 44 —Change EIM Registry
- "eimGetTargetFromSource() —Get EIM Target Identities from the Source" on page 134 —Get EIM Target Identities from the Source
- "eimGetTargetFromIdentifier() —Get EIM Target Identities from the Identifier" on page 128 —Get EIM Target Identities from the Identifier

Example

See Code disclaimer information for information pertaining to code examples.

The following example adds a default registry policy association.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;
    EimPolicyAssociationInfo assocInfo;

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up policy association information    */
    assocInfo.type = EIM_DEFAULT_REG_POLICY;
    assocInfo.policyAssociation.defaultRegistry.sourceRegistry = "MySourceRegistry";
    assocInfo.policyAssociation.defaultRegistry.targetRegistry = "localRegistry";
    assocInfo.policyAssociation.defaultRegistry.targetRegistryUserName = "mjjones";

    /* Add the policy association               */
    if (0 != (rc = eimAddPolicyAssociation(handle,
                                           &assocInfo,
                                           err)))
    {
        printf("Add EIM Policy Association error = %d", rc);
        return -1;
    }

    return 0;

}
```

API introduced: V5R3

Top | Security APIs | APIs by category

---

eimAddPolicyFilter()—Add EIM Policy Filter

Syntax

```
#include <eim.h>

int eimAddPolicyFilter(EimHandle           * eim,
                       EimPolicyFilterInfo * filterInfo,
                       EimRC               * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimAddPolicyFilter()** function adds the specified policy filter value to the domain. A policy association can then be added to the policy filter value using the Add EIM Policy Association (eimAddPolicyAssociation) API. A policy association is used in a mapping lookup operation (eimGetTargetFromSource) if a specific source association does not exist. A policy association to a policy filter value is used to map users with similar attributes to the same target identity in the target registry. You can use the Format EIM Policy Filter (eimFormatPolicyFilter) API to have a policy filter value created for you in the correct format based on the data that is provided.

» EIM version 2 must be supported by the local EIM APIs to use this API (see "eimGetVersion()—Get EIM Version" on page 140—Get EIM Version). «

Warning: Temporary Level 3 Header

Certificate policy filter details

A certificate policy filter is used to map user (or client) certificates with similar attributes to the same target identity in the target registry. For example, a certificate policy filter can be added so that all certificates issued by the same Certificate Authority (CA) are mapped to the same target identity in the target registry. Or, all certificates from the same organization are mapped to the same target identity in the target registry.

To locate a certificate policy filter, a search will be done using a series of full and partial distinguished names (DNs) until the most specific matching filter policy is found. The following values are used in sequence to search for a matching certificate filter policy:

1. *<SDN>subject's-full-DN</SDN><IDN>issuer's-full-DN</IDN>*
   example: <SDN>CN=John D. Smith,OU=Sales,O=IBM,L=Rochester,ST=Min,C=US</SDN><IDN>OU=VeriSign Class 1 Individual Subscriber,O=VeriSign,L=Internet</IDN>
2. *<SDN>subject's-partial-DN</SDN><IDN>issuer's-full-DN</IDN>*
   example: <SDN>O=IBM,L=Rochester,ST=Min,C=US</SDN><IDN>OU=VeriSign Class 1 Individual Subscriber,O=VeriSign,L=Internet</IDN>
3. *<SDN>subject's-full-DN</SDN>*
   example: <SDN>CN=John D. Smith,OU=Sales,O=IBM,L=Rochester,ST=Min,C=US</SDN>
4. *<SDN>subject's-partial-DN</SDN>*
   example: <SDN>OU=Sales,O=IBM,L=Rochester,ST=Min,C=US</SDN>
5. *<IDN>issuer's-full-DN</IDN>*
   example: <IDN>OU=VeriSign Class 1 Individual Subscriber,O=VeriSign,L=Internet</IDN>
6. *<IDN>issuer's-partial-DN</IDN>*
   example: <IDN>O=VeriSign,L=Internet</IDN>

Note that searching is not done for the following values:

- *<SDN>subject's-full-DN</SDN><IDN>issuer's-partial-DN</IDN>*
- *<SDN>subject's-partial-DN</SDN><IDN>issuer's-partial-DN</IDN>*

Each step of the search using a partial DN may actually involve a series of searches for partial name values based on the full DN. Each partial DN value in the series is determined by removing the next most specific node in the DN. The nodes are removed from the most specific to the least specific, in the order that they appear in the DN.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

filterInfo (Input)

The information about the policy filter to be added.

The EimPolicyFilterInfo structure contains information about the policy filter to add.

For EIM_CERTIFICATE_FILTER (1) policy filter type, the *filter* field must contain an EimCertificatePolicyFilter structure. The *sourceRegistry* field must contain the name of a registry that has a type of X.509.

The structure layouts follow:

```
enum EimPolicyFilterType {
    EIM_ALL_FILTERS,               /* All policy filters -- not
                                      supported for this interface.  */
    EIM_CERTIFICATE_FILTER         /* Policy filter is a certificate
                                      filter.                        */
};

typedef struct EimCertificatePolicyFilter
{
    char  * sourceRegistry;        /* The source registry to add the
                                      policy filter to.              */
    char  * filterValue;           /* The policy filter value.       */
} EimCertificatePolicyFilter;

typedef struct EimPolicyFilterInfo
{
    enum EimPolicyFilterType type;
    union {
        EimCertificatePolicyFilter   certFilter;
    } filter;
} EimPolicyFilterInfo;
```

eimrc (Input/Output)

The structure in which to return error code information. If the return value is not 0, eimrc is set with additional information. This parameter may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES

Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry name is not valid 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) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_POLICY_FILTER_TYPE_INVAL (60)	Policy filter type is not valid.
EIMERR_REGTYPE_INVAL (62)	Registry type is not valid.
EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified function is not supported by the EIM version.

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 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.
EIMERR_UNEXP_OBJ_VIOLATION (56) Unexpected object violation.

Related Information

- "eimRemovePolicyFilter()—Remove EIM Policy Filter" on page 232 —Remove EIM PolicyFilter
- "eimListPolicyFilters()—List EIM Policy Filters" on page 166 —List EIM Policy Filters
- "eimFormatPolicyFilter()—Format EIM Policy Filter" on page 78 —Format EIM Policy Filter
- "eimAddPolicyAssociation()—Add EIM Policy Association" on page 26 —Add EIM Policy Association
- "eimRemovePolicyAssociation()—Remove EIM Policy Association" on page 228 —Remove EIM Policy Association
- "eimListRegistryAssociations()—List EIM Registry Associations" on page 184 —List EIM Registry Associations

Example

See Code disclaimer information for information pertaining to code examples.

The following example adds a policy filter.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;
    EimPolicyFilterInfo filterInfo;

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up policy filter information         */
    filterInfo.type = EIM_CERTIFICATE_FILTER;
    filterInfo.filter.certFilter.sourceRegistry = "MySourceRegistry";
    filterInfo.filter.certFilter.filterValue =
      "<IDN>OU=VeriSign Class 1 Individual Subscriber,O=VeriSign,L=Internet</IDN>";

    /* Add the policy filter                    */
    if (0 != (rc = eimAddPolicyFilter(handle,
```

```
                                    &filterInfo,
                                    err)))
    {
        printf("Add EIM Policy Filter error = %d", rc);
        return -1;
    }

    return 0;

}
```

API introduced: V5R3

---

eimChangeDomain()—Change an EIM Domain Object

Syntax

```
#include <eim.h>

int eimChangeDomain(char               * ldapURL,
                    EimConnectInfo       connectInfo,
                    enum EimDomainAttr   attrName,
                    char               * attrValue,
                    enum EimChangeType   changeType,
                    EimRC              * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimChangeDomain()** function changes an attribute for the EIM domain entry identified by ldapURL.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

ldapURL (Input)

A uniform resource locator (URL) that contains the EIM host information. This URL has the following format:

```
ldap://host:port/dn
      or
ldaps://host:port/dn
```

where:

- `host:port` is the name of the host on which the EIM domain controller is running with an optional port number.
- `dn` is the distinguished name of the domain to change.
- `ldaps` indicates that this host/port combination uses SSL and TLS.

Examples:

- ldap://systemx:389
- ldaps://systemy:636/o=ibm,c=us

connectInfo (Input)

Connect information. EIM uses ldap. 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 EIM_SIMPLE connect type, the `creds` field should contain the EimSimpleConnectInfo structure with a binddn and password. EimPasswordProtect is used to determine 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. EimSSLInfo must be provided.

The structure layouts follow:

```
enum EimPasswordProtect {
    EIM_PROTECT_NO,
    EIM_PROTECT_CRAM_MD5,
    EIM_PROTECT_CRAM_MD5_OPTIONAL
};
enum EimConnectType {
    EIM_SIMPLE,
    EIM_KERBEROS,
    EIM_CLIENT_AUTHENTICATION
};

typedef struct EimSimpleConnectInfo
{
     enum EimPasswordProtect protect;
     char * bindDn;
     char * bindPw;
} EimSimpleConnectInfo;

typedef struct EimSSLInfo
{
     char * keyring;
     char * keyring_pw;
     char * certificateLabel;
} EimSSLInfo;

typedef struct EimConnectInfo
{
     enum EimConnectType type;
     union {
         gss_cred_id_t * kerberos;
         EimSimpleConnectInfo simpleCreds;
     } creds;
   EimSSLInfo * ssl;
} EimConnectInfo;
```

attrName (Input)

The attribute to be updated. Valid values are:

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

EIM_DOMAIN_POLICY_ASSOCIATIONS (1) Change the indicator for whether or not the domain supports policy associations in a mapping lookup. By default, the policy associations are not supported. Valid *changeType* is EIM_ENABLE (3) or EIM_DISABLE (4). This attribute is controlled by the version of the API interface, not the domain. If this attribute is enabled for the domain, it will only be checked in a mapping lookup operation if the version of the mapping lookup API that is used to access the domain supports this attribute. » EIM version 2 must be supported by the local EIM APIs to specify this attribute (see "eimGetVersion()—Get EIM Version" on page 140—Get EIM Version). «

attrValue (Input)
The new value for the attribute.

If the attribute being changed is EIM_DOMAIN_POLICY_ASSOCIATIONS, this value must be NULL.

changeType (Input)
The type of change to make. This could be add, remove, change, enable, or disable. *attrName* parameter indicates which type is allowed for each attribute.

eimrc (Input/Output)
The structure in which to return error code information. If the return value is not 0, eimrc will be set with additional information. This parameter may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

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

EIMERR_NODOMAIN (24) EIM Domain not found or insufficient access to EIM data.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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_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 API documentation.
EIMERR_PROTECT_INVAL (22)	The protect parameter in EimSimpleConnectInfo is not valid.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_SSL_REQ (42)	The 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_INVALID_DN (66)	Distinguished Name (DN) is not valid.
EIMERR_FUNCTION_NOT_ SUPPORTED (70)	The specified function is not supported by the EIM version.

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 only be made to a replica ldap server. Change the connection information and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23)	Unexpected LDAP error. %s
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimDeleteDomain()—Delete an EIM Domain Object" on page 72—Delete an EIM Domain Object
- "eimCreateDomain()—Create an EIM Domain Object" on page 65—Create an EIM Domain Object
- "eimListDomains()—List EIM Domain Objects" on page 154—List EIM Domain Objects
- "eimChangeRegistry()—Change EIM Registry" on page 44—Change EIM Registry

Example

See Code disclaimer information for information pertaining to code examples.

The following example changes the description of the specified EIM domain and enables the use of policy associations for the domain.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    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,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* 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);

    /* Enable the use of policy associations. */
    if (0 != (rc = eimChangeDomain(ldapURL,
                                   con,
                                   EIM_DOMAIN_POLICY_ASSOCIATIONS,
                                   NULL,
                                   EIM_ENABLE,
                                   err)))
        printf("Change domain error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

eimChangeIdentifier()— Change EIM Identifier

Syntax

```
#include <eim.h>

int eimChangeIdentifier(EimHandle              * eim,
                        EimIdentifierInfo      * idName,
                        enum EimIdentifierAttr   attrName,
                        char                   * attrValue,
                        enum EimChangeType       changeType,
   EimRC                  * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimChangeIdentifier()** function modifies an existing EIM identifier.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

idName (Input)

A structure that contains the name for this identifier. 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` will indicate which identifier name has been provided. Use of the `uniqueName` will provide the best performance. There is no guarantee that `name` will find a unique identifier. Therefore, use of `name` may result in an error.

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* is

- 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* is

- EIM_ADD (1)
- EIM_RMV (2)

attrValue (Input)
The new value for the attribute.

changeType (Input)
The type of change to make. This could be add, remove, or change. *attrName* parameter indicates which type is allowed for each attribute.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Identifier name is not valid or insufficient access to EIM data.

EIMERR_IDNAME_AMBIGUOUS (20) More than 1 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) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35) Pointer parameter is not valid.

ENOMEM
Unable to allocate required space.

EIMERR_NOMEM (27) No memory available. Unable to allocate required space.

ENOTCONN
LDAP connection has not been made.

EIMERR_NOT_CONN (31) Not connected to LDAP. Use 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.

Restrictions

There is a restriction on the characters allowed for identifier name.

The following characters are special characters that are not allowed in object names. They also should not be used in object attributes that would be used for a search operation.

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

Related Information

- "eimAddIdentifier()—Add EIM Identifier" on page 23—Add EIM Identifier
- "eimRemoveIdentifier()— Remove EIM Identifier" on page 224—Change EIM Identifier
- "eimListIdentifiers()— List EIM Identifiers" on page 160—List EIM Identifiers
- "eimGetAssociatedIdentifiers() —Get Associated EIM identifiers" on page 91 —Get Associated EIM Identifiers

Example

See Code disclaimer information for information pertaining to code examples.

The following example will change an EIM identifier description.

```
#include <eim.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;
    EimIdentifierInfo idInfo;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Set up identifier information          */
    idInfo.idtype = EIM_UNIQUE_NAME;
    idInfo.id.uniqueName = "Mary Smith";

    /* Change the description of the identifier */
    if (0 != (rc = eimChangeIdentifier(handle,
                                       &idInfo,
                                       EIM_IDENTIFIER_DESCRIPTION,
                                       "This is a new description",
                                       EIM_CHG,
                                       err)))
        printf("Change identifier error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimChangeRegistry()—Change EIM Registry

Syntax

```
#include <eim.h>

int eimChangeRegistry(EimHandle           * eim,
                      char                * registryName,
```

```
enum EimRegistryAttr   attrName,
char                 * attrValue,
enum EimChangeType     changeType,
EimRC                * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimChangeRegistry()** function changes the attribute of a registry participating in the EIM domain.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM authority to an individual registry

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

registryName (Input)

The name of the registry to change.

attrName (Input)

The attribute to be updated. Valid values are:

EIM_REGISTRY_DESCRIPTION (0)	Change the registry description. Valid *changeType* is EIM_CHG (0).
EIM_REGISTRY_LABELEDURI (1)	Change the URI for the system registry. Valid *changeType* is EIM_CHG (0).
EIM_REGISTRY_MAPPING_LOOKUP (2)	Change the indicator for whether or not the registry supports any mapping lookup operations. By default, the mapping lookup operations are supported. Valid *changeType* is EIM_ENABLE (3) or EIM_DISABLE (4). This attribute is controlled by the version of the API interface, not the domain. If this attribute is disabled for the registry, it will only be checked in a mapping lookup operation if the version of the mapping lookup API that is used to access the domain supports this attribute. » EIM version 2 must be supported by the local EIM APIs to specify this attribute (see "eimGetVersion()—Get EIM Version" on page 140—Get EIM Version). «

EIM_REGISTRY_POLICY_ASSOCIATIONS (3)
Change the indicator for whether or not the registry supports policy associations in a mapping lookup. By default, the policy associations are not supported. Valid *changeType* is EIM_ENABLE (3) or EIM_DISABLE (4). This attribute is controlled by the version of the API interface, not the domain. If this attribute is enabled for the registry, it will only be checked in a mapping lookup operation if the version of the mapping lookup API that is used to access the domain supports this attribute. » EIM version 2 must be supported by the local EIM APIs to specify this attribute (see "eimGetVersion()—Get EIM Version" on page 140—Get EIM Version). «

» *EIM_REGISTRY_MEMBER (4)*
Change the list of registries that are members of this group registry. The *registryName* parameter must be the name of a registry that has a type of group registry to change this attribute. The registry name specified in the *attrValue* parameter must exist and cannot have a type of group registry. The normalization method for the group registry must be the same as the normalization method for the member. Valid *changeType* is EIM_ADD (1) or EIM_RMV (2). EIM version 3 must be supported by the local EIM APIs to specify this attribute (see "eimGetVersion()—Get EIM Version" on page 140—Get EIM Version). «

attrValue (Input)
The new value for the attribute.

If the attribute being changed is EIM_REGISTRY_MAPPING_LOOKUP or EIM_REGISTRY_POLICY_ASSOCIATIONS, this value must be NULL.

changeType (Input)
The type of change to make. This could be add, remove, change, enable, or disable. *attrName* parameter indicates which type is allowed for each attribute.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry not found or insufficient access to EIM data.

EIMERR_NOREG (28) EIM Registry not found or insufficient access to EIM data.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_REGTYPE_INVAL (62)	Registry type is not valid.
EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified function is not supported by the EIM version.

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 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.

Related Information

- "eimAddSystemRegistry()—Add a System Registry to the EIM domain" on page 3 —Add a System Registry to the EIM Domain
- "eimAddApplicationRegistry()—Add an Application Registry to the EIM Domain" on page 7 —Add an Application Registry to the EIM Domain
- "eimAddGroupRegistry()—Add a Group Registry to the EIM domain" on page 19 —Add a Group Registry to the EIM Domain
- "eimRemoveRegistry()—Remove a Registry from the EIM Domain" on page 214 —Remove a Registry from the EIM Domain
- "eimListRegistries()—List EIM Registries" on page 171 —List EIM Registries
- "eimChangeDomain()—Change an EIM Domain Object" on page 36 —Change EIM Domain

Example

See Code disclaimer information for information pertaining to code examples.

The following example changes the description for the registry and enables the use of policy associations for the registry.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                 */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Change the description for this registry */
    if (0 != (rc = eimChangeRegistry(handle,
                                     "MyAppRegistry",
                                     EIM_REGISTRY_DESCRIPTION,
                                     "New description",
                                     EIM_CHG,
                                     err)))
        printf("Change registry error = %d", rc);

    /* Enable the use of default registry policies. */
    if (0 != (rc = eimChangeRegistry(handle,
                                     "MyAppRegistry",
                                     EIM_REGISTRY_POLICY_ASSOCIATIONS,
                                     NULL,
```

```
                                    EIM_ENABLE,
                                    err)))
        printf("Change registry error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimChangeRegistryAlias()—Change EIM Registry Alias

Syntax

```
#include <eim.h>

int eimChangeRegistryAlias(EimHandle          * eim,
                           char               * registryName,
                           char               * aliasType,
                           char               * aliasValue,
                           enum EimChangeType   changeType,
                           EimRC              * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimChangeRegistryAlias()** function allows you to add or remove a registry alias for the defined registry.

One way to decouple names used by developers and names chosen by administrators is by using registry aliases. When designing applications, developers know the registry type their application uses and choose the registry alias their program will use. Developers communicate to the administrator which registry types their applications depend on along with the EIM registry aliases that must be associated with that registry type. The administrator adds the registry alias to the EIM registry of the appropriate type. The application can use eimGetRegistryNameFromAlias() API which, given a registry alias, returns the registry name for the entry(ies) with that registry alias.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM authority to this individual registry

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

registryName (Input)

The name of the registry to be changed.

aliasType (Input)
A type of alias for this registry. The user may supply their own alias type. There is a list of predefined alias types in eim.h.

aliasValue (Input)
The value for this alias.

changeType (Input)
The type of change to make. This could be add or remove.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry not found or insufficient access to EIM data.

EIMERR_NOREG (28) EIM Registry not found or insufficient access to EIM data.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.

EIMERR_PTR_INVAL (35) Pointer parameter is not valid.

ENOMEM
Unable to allocate required space.

EIMERR_NOMEM (27) No memory available. Unable to allocate required space.

ENOTCONN
LDAP connection has not been made.

EIMERR_NOT_CONN (31) Not connected to LDAP. Use 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.

Restrictions

The wild card character (*) should not be used for registry aliases.

Related Information

- "eimListRegistryAliases()—List EIM Registry Aliases" on page 179 —List EIM Registry Aliases
- "eimGetRegistryNameFromAlias() —Get EIM Registry Name from an Alias" on page 109 —Get EIM Registry Name from an Alias

Example

See Code disclaimer information for information pertaining to code examples.

The following example adds a couple aliases to the registry.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];
```

```
    /* Set up error structure.                */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Add a dns alias for this registry      */
    if (0 != (rc = eimChangeRegistryAlias(handle,
                                          "MyRegistry",
                                          EIM_ALIASTYPE_DNS,
                                          "Clueless",
                                          EIM_ADD,
                                          err)))
    {
        printf("Change registry alias error = %d", rc);
        return -1;
    }
    /* Add a tcpip address as an alias        */
    if (0 != (rc = eimChangeRegistryAlias(handle,
                                          "MyRegistry",
                                          EIM_ALIASTYPE_TCPIP,
                                          "9.5.2.12",
                                          EIM_ADD,
                                          err)))
    {
        printf("Change registry alias error = %d", rc);
        return -1;
    }

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimChangeRegistryUser() —Change EIM Registry User

Syntax

```
#include <eim.h>

int eimChangeRegistryUser(EimHandle               * eim,
                          char                    * registryName,
                          char                    * registryUserName,
                          enum EimRegistryUserAttr  attrName,
                          char                    * attrValue,
                          enum EimChangeType        changeType,
        EimRC                    * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimChangeRegistryUser()** function changes the attributes of a registry user entry. A registry user is implicitly added to a registry when a target association for an identity in that registry is added. However, the attribute fields are not set at that time.

There are situations when more than one user can be returned on a mapping lookup operation. Applications can choose to use information in the additional information field to distinguish between which returned target identity to use. For example, assume 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().

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM authority to an individual registry

» Must be a member of EIM Administrator or have EIM authority to an individual registry to change the EIM_REGUSER_PASSWORD_CRED (2), EIM_REGUSER_PWD_CRED_STATUS (3), and EIM_REGUSER_IDCTX_CRED (4) attributes. «

» Note that if the registry existed prior to EIM Version 3, the first time credential information is added to a user in that registry, it must be added by a member of EIM Administrator. After the first credential information is added, then EIM authority to an individual registry is sufficient to change or remove credential information for any user in the registry. «

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

registryName (Input)

The name of the registry that contains this user.

registryUserName (Input)

The name of the user in this registry to change.

attrName

The attribute to be updated. 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. Valid *changeType* is EIM_ADD (1) and EIM_RMV (2).
» *EIM_REGUSER_PASSWORD_CRED (2)*	Change the password credential associated with the registry user. Valid *changeType* is EIM_CHG (0) and EIM_RMV (2). EIM version 3 must be supported by the local EIM APIs to specify this attribute (see “eimGetVersion()—Get EIM Version” on page 140—Get EIM Version). «
» *EIM_REGUSER_PWD_CRED_STATUS (3)*	Change the status of the password credential associated with the registry user. Valid *changeType* is EIM_ENABLE (3) and EIM_DISABLE (4). This attribute has no affect if the EIM_REGUSER_PASSWORD_CRED (2) attribute does not exist for the registry user. EIM version 3 must be supported by the local EIM APIs to specify this attribute (see “eimGetVersion()—Get EIM Version” on page 140—Get EIM Version). «

EIM_REGUSER_IDCTX_CRED (4) — Change the identity context credential associated with the registry user. Valid *changeType* is EIM_CHG (0) and EIM_RMV (2). EIM version 3 must be supported by the local EIM APIs to specify this attribute (see "eimGetVersion()—Get EIM Version" on page 140—Get EIM Version).

attrValue (Input)
The new value for the attribute.

If the attribute being changed is EIM_REGUSER_PASSWORD_CRED or EIM_REGUSER_IDCTX_CRED and the *changeType* is EIM_RMV (2), or the attribute being changed is EIM_REGUSER_PWD_CRED_STATUS, this value must be NULL.

If the attribute being changed is EIM_REGUSER_IDCTX_CRED and the *changeType* is EIM_CHG (0), then this value must be a pointer to an EimBinaryData structure. The layout of the EimBinaryData structure follows:

```
typedef struct EimBinaryData
{
    int             length;
    unsigned char  * data;
} EimBinaryData;
```

changeType (Input)
The type of change to make. This could be add, remove, change, enable, or disable. *attrName* parameter indicates which type is allowed for each attribute.

eimrc (Input/Output)
The structure in which to return error code information. If the return value is not 0, eimrc will be set with additional information. This parameter may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) — Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry or registry user 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) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified function is not supported by the EIM version.

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 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_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.

EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimListRegistryUsers()— List EIM Registry Users" on page 191—List EIM Registry Users

Example

See Code disclaimer information for information pertaining to code examples.

The following example changes the description and adds additional information for the target registry user.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                 */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Change the registry user's description      */
    if (0 != (rc = eimChangeRegistryUser(handle,
                                         "MyRegistry",
                                         "mjjones",
                                         EIM_REGISTRYUSER_DESCRIPTION,
                                         "cool customer",
                                         EIM_CHG,
                                         err)))
    {
        printf("Change registry user error = %d", rc);
        return -1;
    }

    /* Add additional information to the registry user*/
    if (0 != (rc = eimChangeRegistryUser(handle,
                                         "MyRegistry",
                                         "mjjones",
                                         EIM_REGISTRYUSER_ADDL_INFO,
                                         "security officer",
                                         EIM_ADD,
                                         err)))
    {
        printf("Change registry user error = %d", rc);
        return -1;
    }

    /* Add additional information to the registry user*/
    if (0 != (rc = eimChangeRegistryUser(handle,
                                         "MyRegistry",
                                         "mjjones",
                                         EIM_REGISTRYUSER_ADDL_INFO,
                                         "administrator",
                                         EIM_ADD,
```

```
                                        err)))
    {
        printf("Change registry user error = %d", rc);
        return -1;
    }

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimConnect()—Connect to EIM Domain

Syntax

```
#include <eim.h>

int eimConnect(EimHandle      * eim,
               EimConnectInfo   connectInfo,
   EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimConnect()** function is used to connect to the EIM domain that is configured for this platform. Configuration information was set using eimSetConfiguration().

Authorities and Locks

None.

Parameters

eim (Input)
The EIM handle returned by a previous call to eimCreateHandle().

connectInfo (Input)
Connect information. EIM uses ldap. 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 EIM_SIMPLE connect type, the `creds` field should contain the EimSimpleConnectInfo structure with a binddn and password. EimPasswordProtect is used to determine 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. EimSSLInfo must be provided.

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, eimrc is set with additional information. This parameter may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PROTECT_INVAL (22)	The protect parameter in EimSimpleConnectInfo is not valid.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_SSL_REQ (42)	The system is configured to connect to a secure port. EimSSLInfo is required.

EISCONN
A connection has already been established.

EIMERR_CONN (11)	Connection already exists.

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. %s
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimCreateHandle()—Create an EIM Handle" on page 70—Create an EIM Handle
- "eimDestroyHandle()—Destroy an EIM Handle" on page 76—Destroy an EIM Handle
- "eimGetAttribute()—Get EIM attributes" on page 98—Get EIM Attributes
- "eimSetAttribute()—Set EIM attributes" on page 239—Set EIM Attributes
- "eimConnectToMaster()—Connect to EIM Master Domain" on page 60—Connect to EIM Master Domain

Example

See Code disclaimer information for information pertaining to code examples.

The following example will connect to an EIM domain.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    EimConnectInfo con;

    /* Get eim handle from input arg.           */
    /* This handle should not be connected to   */
    /* the configuration system.                */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                 */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* 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 configuartion system         */
    if (0 != (rc = eimConnect(handle,
                              con,
                              err)))
        printf("Connect error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimConnectToMaster()—Connect to EIM Master Domain

Syntax

```
#include <eim.h>

int eimConnectToMaster(EimHandle      * eim,
                       EimConnectInfo   connectInfo,
                       EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimConnectToMaster()** function is used to connect to the EIM master domain controller. This API should be used if an earlier API invocation returned a referral error (EROFS). A referral error indicates that the current EIM connection is to a replication system. An explicit connection must be made to the master system in order to make updates.

The ldap configuration file is used to retrieve information for the master host, master port, and secure port. 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.

Authorities and Locks

None.

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle().

connectInfo (Input)

Connect information. EIM uses ldap. 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 EIM_SIMPLE connect type, the `creds` field should contain the EimSimpleConnectInfo structure with a binddn and password. EimPasswordProtect is used to determine 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. EimSSLInfo must be provided.

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, eimrc is set with additional information. This parameter may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PROTECT_INVAL (22)	The protect parameter in EimSimpleConnectInfo is not valid.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_SSL_REQ (42)	The system is configured to connect to a secure port. EimSSLInfo is required.

EISCONN
A connection has already been established.

EIMERR_CONN (11) Connection already exists.

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. %s
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimCreateHandle()—Create an EIM Handle" on page 70—Create an EIM Handle
- "eimDestroyHandle()—Destroy an EIM Handle" on page 76—Destroy an EIM Handle
- "eimGetAttribute()—Get EIM attributes" on page 98—Get EIM Attributes
- "eimSetAttribute()—Set EIM attributes" on page 239—Set EIM Attributes
- "eimConnect()—Connect to EIM Domain" on page 57—Connect to EIM Domain

Example

See Code disclaimer information for information pertaining to code examples.

The following example will connect to an EIM master domain.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    EimConnectInfo con;

    /* Get eim handle from input arg.           */
    /* This handle should not be connected to   */
    /* the master system.                       */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                 */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;
```

```
    /* 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 system.               */
    if (0 != (rc = eimConnectToMaster(handle,
                                      con,
                                      err)))
        printf("Connect error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimErr2String()—Convert EimRC into an Error Message

Syntax

```
#include <eim.h>

char * eimErr2String(EimRC              * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimErr2String()** function converts the EIM return code structure returned by an EIM function into a NULL-terminated error message string. free() should be used to free the space allocated for the error message string.

Authorities

No authorization is required.

Parameters

eimrc (Input)
The structure that contains error code information from a previous call to an EIM API. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

If successful, the return value is the address of the error message. The caller is responsible for freeing the message.

If unsuccessful, eimErr2String returns a NULL pointer. The *errno* global variable is set to indicate the error. The errno may come from catopen, catget, or catclose or one of the following values.

EBADDATA
eimrc is not valid.

ECONVERT
Data conversion error.

EINVAL
Input parameter was not valid.

ENOMEM
Unable to allocate required space.

EUNKNOWN
Unexpected exception.

Related Information

- "EimRC—EIM Return Code" on page 254 —EIM Return Code Parameter

Example

See Code disclaimer information for information pertaining to code examples.

The following example converts an EimRC into an error message and prints it.

```
#include <eim.h>
#include <stdio.h>
#include <errno.h>

int main(int argc, char *argv[])
{
    int           rc;
    EimRC       * err;
    char        * errMessage;

    /* Get EimRC from input arg.                */
    err = (EimRC *)argv[1];

    /* Get error message                       */
    if (NULL == (errMessage = eimErr2String(err)))
    {
        printf("eimErr2String error = %s", strerror(errno));
        return -1;
    }

    /* Print the message                       */
    printf("%s", errMessage);

    free(errMessage);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimCreateDomain()—Create an EIM Domain Object

Syntax

```
#include <eim.h>

int eimCreateDomain(char           * ldapURL,
                    EimConnectInfo   connectInfo,
                    char           * description,
        EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimCreateDomain()** function creates an EIM domain object on the specified EIM domain controller.

Authorities and Locks

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

Parameters

ldapURL (Input)
A uniform resource locator (URL) that contains the EIM host information. This URL has the following format:

```
ldap://host:port/dn
      or
ldaps://host:port/dn
```

where:

- `host:port` is the name of the host on which the EIM domain controller is running with an optional port number.
- `dn` is the distinguished name of the domain to create.
- `ldaps` indicates that this host/port combination uses SSL and TLS.

Examples:

- ldap://systemx:389/ibm-eimDomainName=myEimDomain,o=myCompany,c=us
- ldaps://systemy:636/ibm-eimDomainName=thisEimDomain

connectInfo (Input)
Connect information. EIM uses ldap. 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 EIM_SIMPLE connect type, the `creds` field should contain the EimSimpleConnectInfo structure with a binddn and password. EimPasswordProtect is used to determine 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. EimSSLInfo must be provided.

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 may be NULL.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EEXIST
EIM domain already exists.

EIMERR_DOMAIN_EXISTS (14) EIM domain already exists in EIM.

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_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 API documentation.
EIMERR_PROTECT_INVAL (22)	The protect parameter in EimSimpleConnectInfo is not valid.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_SSL_REQ (42)	The 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_INVALID_DN (66)	Distinguished Name (DN) is not valid.

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 only be made to a replica ldap server. Change the connection information and try the request again.

EUNKNOWN

Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s

EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Restrictions

There is a restriction on the characters allowed for domain name.

The following characters are special characters that are not allowed in object names. They also should not be used in object attributes that would be used for a search operation.

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

Related Information

- "eimDeleteDomain()—Delete an EIM Domain Object" on page 72—Delete an EIM Domain Object
- "eimChangeDomain()—Change an EIM Domain Object" on page 36—Change an EIM Domain Object
- "eimListDomains()—List EIM Domain Objects" on page 154—List EIM Domain Objects

Example

See Code disclaimer information for information pertaining to code examples.

The following example creates an EIM domain by 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>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    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,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Create a new EIM domain                 */
    if (0 != (rc = eimCreateDomain(ldapURL,
                                   con,
                                   NULL,
                                   err)))
        printf("Create domain error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

eimCreateHandle()—Create an EIM Handle

Syntax

```
#include <eim.h>

int eimCreateHandle(EimHandle      * eim,
                    char           * ldapURL,
      EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimCreateHandle()** function is used to allocate an EimHandle structure, which is used to identify the EIM connection and to maintain per-connection information. The EimHandle structure should be passed on subsequent calls to other EIM operations.

Authorities and Locks

If a NULL is not passed for the ldapURL parameter, then the caller of the API must have *SECADM special authority.

Parameters

eim (Output)
The pointer to an EIM handle to be returned. This handle is used as input for other EIM APIs. The handle is temporary; you can use it only in the job that created it.

ldapURL (Input)
A uniform resource locator (URL) that contains the EIM host information. A NULL parameter indicates that the `ldapURL` information set by the eimSetConfiguration() API should be used. This URL has the following format:

```
ldap://host:port/dn
      or
ldaps://host:port/dn
```

where:

- `host:port` is the name of the host on which the EIM domain controller is running with an optional port number.
- `dn` is the distinguished name of the domain to work with.
- `ldaps` indicates that this host/port combination uses SSL and TLS.

Examples:

- ldap://systemx:389/ibm-eimDomainName=myEimDomain,o=myCompany,c=us
- ldaps://systemy:636/ibm-eimDomainName=thisEimDomain

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied.

EIMERR_AUTH_ERR (7) Insufficient authority for the operation.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EINVAL
Input parameter was not valid.

EIMERR_PARM_REQ (34) Missing required parameter. Please check 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.
EIMERR_INVALID_DN (66) Distinguished Name (DN) is not valid.

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. Run eimSetConfiguration() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimDestroyHandle()—Destroy an EIM Handle" on page 76—Destroy an EIM Handle
- "eimGetAttribute()—Get EIM attributes" on page 98—Get EIM Attributes
- "eimSetAttribute()—Set EIM attributes" on page 239—Set EIM Attributes
- "eimConnectToMaster()—Connect to EIM Master Domain" on page 60—Connect to EIM Master Domain
- "eimConnect()—Connect to EIM Domain" on page 57—Connect to EIM Domain

Example

See Code disclaimer information for information pertaining to code examples.

The following example creates an EIM handle.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle     handle;
    EimHandle     handle2;
    char * ldapURL = "ldap://eimsystem:389/ibm-eimDomainName=myEimDomain,o=mycompany,c=us";

    /* Set up error structure.                */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;


    /* Create a new eim handle. Use the eim configuration URL */
    if (0 != (rc = eimCreateHandle(&handle,
                                   NULL,
                                   err)))
        printf("Create handle error = %d", rc);

    /* Create a new eim handle. Use the specified URL */
    if (0 != (rc = eimCreateHandle(&handle2,
                                   ldapURL,
                                   err)))
        printf("Create handle error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimDeleteDomain()—Delete an EIM Domain Object

Syntax

```
#include <eim.h>

int eimDeleteDomain(char           * ldapURL,
                    EimConnectInfo   connectInfo,
        EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimDeleteDomain()** function deletes the EIM domain information. If there are any registries or identifiers in the domain then it cannot be deleted.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

ldapURL (Input)

A uniform resource locator (URL) that contains the EIM host information. This URL has the following format:

```
ldap://host:port/dn
      or
ldaps://host:port/dn
```

where:

- `host:port` is the name of the host on which the EIM domain controller is running with an optional port number.
- `dn` is the distinguished name of the domain to delete.
- `ldaps` indicates that this host/port combination uses SSL and TLS.

Examples:

- ldap://systemx:389/ibm-eimDomainName=myEimDomain,o=myCompany,c=us
- ldaps://systemy:636/ibm-eimDomainName=thisEimDomain

connectInfo (Input)

Connect information. EIM uses ldap. 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 EIM_SIMPLE connect type, the `creds` field should contain the EimSimpleConnectInfo structure with a binddn and password. EimPasswordProtect is used to determine 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. EimSSLInfo must be provided.

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, eimrc will be set with additional information. This parameter may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

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

EIMERR_NODOMAIN (24) EIM Domain not found or insufficient access to EIM data.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EINVAL
Input parameter was not valid.

EIMERR_CONN_INVAL (54) Connection type is not valid.
EIMERR_NOT_SECURE (32) The system is not configured to connect to a secure port. Connection type of EIM_CLIENT_AUTHENTICATION is not valid.
EIMERR_PARM_REQ (34) Missing required parameter. Please check API documentation.
EIMERR_PROTECT_INVAL (22) The protect parameter in EimSimpleConnectInfo is not valid.
EIMERR_PTR_INVAL (35) Pointer parameter is not valid.
EIMERR_SSL_REQ (42) The 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_INVALID_DN (66) Distinguished Name (DN) is not valid.

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 only be made to a replica ldap server. Change the connection information and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s

EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimCreateDomain()—Create an EIM Domain Object" on page 65—Create an EIM Domain Object
- "eimChangeDomain()—Change an EIM Domain Object" on page 36—Change an EIM Domain Object
- "eimListDomains()—List EIM Domain Objects" on page 154—List EIM Domain Objects

Example

See Code disclaimer information for information pertaining to code examples.

The following example deletes the specified EIM domain information.

```
#include <eim.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    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,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;


    /* Delete this domain                      */
    if (0 != (rc = eimDeleteDomain(ldapURL,
                                   con,
                                   err)))
        printf("Delete domain error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimDestroyHandle()—Destroy an EIM Handle

Syntax

```
#include <eim.h>

int eimDestroyHandle(EimHandle      * eim,
         EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimDestroyHandle()** function is used to deallocate an EimHandle structure. This will close any EIM connections for this handle.

Authorities and Locks

None.

Parameters

eim (Input)
The EIM handle returned by a previous call to eimCreateHandle().

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

EINVAL
Input parameter was not valid.

EIMERR_HANDLE_INVAL (17) EimHandle is not valid.
EIMERR_PARM_REQ (34) Missing required parameter. Please check API documentation.
EIMERR_PTR_INVAL (35) Pointer parameter is not valid.

EUNKNOWN
Unexpected exception.

EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimCreateHandle()—Create an EIM Handle" on page 70—Create an EIM Handle
- "eimGetAttribute()—Get EIM attributes" on page 98—Get EIM Attributes

- "eimSetAttribute()—Set EIM attributes" on page 239—Set EIM Attributes
- "eimConnectToMaster()—Connect to EIM Master Domain" on page 60—Connect to EIM Master Domain
- "eimConnect()—Connect to EIM Domain" on page 57—Connect to EIM Domain

Example

See Code disclaimer information for information pertaining to code examples.

The following example destroys an EIM handle.

```
#include <eim.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;


    /* Get eim handle from input arg.           */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Destroy the handle                       */
    if (0 != (rc = eimDestroyHandle(handle,
                                    err)))
        printf("Destroy handle error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimFormatPolicyFilter()—Format EIM Policy Filter

Syntax

```
#include <eim.h>

int eimFormatPolicyFilter(EimUserIdentityInfo         * userIdentityInfo,
                          EimPolicyFilterSubsetInfo   * subsetInfo,
                          unsigned int                  lengthOfListData,
                          EimList                     * listData,
                          EimRC                       * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimFormatPolicyFilter()** function takes unformatted user identity information and generates a policy filter value for use with the Add EIM Policy Filter (eimAddPolicyFilter) API.

Warning: Temporary Level 3 Header

Certificate policy filter details

A formatted certificate policy filter value will contain a combination of the subject and issuer full and partial distinguished names (DNs). The following are the different combinations that can be generated, based on the *subjectFilter* and *issuerFilter* values:

- *<SDN>subject's-full-DN</SDN><IDN>issuer's-full-DN</IDN>*
- *<SDN>subject's-partial-DN</SDN><IDN>issuer's-full-DN</IDN>*
- *<SDN>subject's-full-DN</SDN>*
- *<SDN>subject's-partial-DN</SDN>*
- *<IDN>issuer's-full-DN</IDN>*
- *<IDN>issuer's-partial-DN</IDN>*

Note that the following combinations can be generated, but would never be used when doing a mapping lookup:

- *<SDN>subject's-full-DN</SDN><IDN>issuer's-partial-DN</IDN>*
- *<SDN>subject's-partial-DN</SDN><IDN>issuer's-partial-DN</IDN>*

Specifying a value for the *subjectFilter* or *issuerFilter* fields in the *subsetInfo* parameter will determine the content of the policy filter. If the *subsetInfo* parameter is NULL, then all possible valid certificate policy filter values will be returned.

Specifying a value for the *subjectFilter* field indicates that the subject DN information should be included in the policy filter value, and where in the DN to start generating the subject DN value. For example, specifying "OU=" for the *subjectFilter* field will start generating the subject DN portion of the value with the OU node (will not include any nodes before the OU node). Specifying a value for the *issuerFilter* field indicates that the issuer DN information should be included in the policy filter value, and where in the DN to start generating the issuer DN value. If NULL is specified for both the *subjectFilter* field and the *issuerFilter* field, the policy filter value will contain the subject's full DN and the issuer's full DN.

When using the issuer or subject filter value, the value specified in the filter value must exist in the DN.

Given a certificate, where the subject DN is:

```
CN=John D. Smith,OU=Sales,O=IBM,L=Rochester,ST=Min,C=US
```

and the issuer DN is:

```
OU=VeriSign Class 1 Individual Subscriber,O=VeriSign,L=Internet
```

- if a value of "OU=" is specified for the *subjectFilter* field and NULL is specified for the *issuerFilter* field, the following policy filter value will be generated:
  <SDN>OU=SALES,O=IBM,L=ROCHESTER,ST=MIN,C=US</SDN>
- if a value of "OU=" is specified for the *subjectFilter* field and "OU=" is specified for the *issuerFilter* field, the following policy filter value will be generated:
  <SDN>OU=SALES,O=IBM,L=ROCHESTER,ST=MIN,C=US</SDN><IDN>OU=VERISIGN CLASS 1 INDIVIDUAL SUBSCRIBER,O=VERISIGN,L=INTERNET</IDN>
- if a value of NULL is specified for the *subjectFilter* field and "OU=" is specified for the *issuerFilter* field, the following policy filter value will be generated:
  <IDN>OU=VERISIGN CLASS 1 INDIVIDUAL SUBSCRIBER,O=VERISIGN,L=INTERNET</IDN>
- if a value of NULL is specified for the *subjectFilter* field and "O=" is specified for the *issuerFilter* field, the following policy filter value will be generated:
  <IDN>O=VERISIGN,L=INTERNET</IDN>
- if a value of NULL is specified for the *subjectFilter* field and the *issuerFilter* field, the following policy filter value will be generated:

<SDN>CN=JOHN D.
SMITH,OU=SALES,O=IBM,L=ROCHESTER,ST=MIN,C=US</SDN><IDN>OU=VERISIGN CLASS 1
INDIVIDUAL SUBSCRIBER,O=VERISIGN,L=INTERNET</IDN>

NOTE: EIM recognizes all of the suggested naming attributes from RFC 3280 with a few additions. They are defined in the following table. If EIM encounters a naming attribute in a certificate that it does not recognize, the OID for the naming attribute will be used instead in the filter value. If you are using the eimCertificateInfo structure, the OID value for any naming attribute that is not in this table may be used.

OID	Naming Attribute	Description
2.5.4.6	c	This attribute contains a two-letter ISO 3166 country or region code (countryName). RFC 3280.
2.5.4.3	cn	This is the X.500 commonName attribute, which contains a name of an object. If the object corresponds to a person, it is typically the persons full name. RFC 3280.
0.9.2342.19200300.100.1.25	dc	Specifies one component of a domain name. RFC 3280.
2.5.4.46	dnQualifier	The dnQualifier attribute type specifies disambiguating information to add to the relative distinguished name of an entry. It is intended for use when merging data from multiple sources in order to prevent conflicts between entries which would otherwise have the same name. It is recommended that the value of the dnQualifier attribute be the same for all entries from a particular source. RFC3280.
1.2.840.113549.1.9.1	email	E-mail address
2.5.4.44	generationQualifier	Contains the part of the name which typically is the suffix, as in IIIrd. RFC 3280.
2.5.4.42	givenName	Used to hold the part of a persons name which is not their surname nor middle name. RFC 3280.
2.5.4.43	initials	The initials attribute contains the initials of some or all of an individuals names, but not the surname(s). RFC 3280.
2.5.4.7	l	This attribute contains the name of a locality, such as a city, county or other geographic region (localityName). RFC 3280.
0.9.2342.19200300.100.1.3	mail	Identifies a user's primary e-mail address (the e-mail address retrieved and displayed by ″white-pages″ lookup applications).
2.5.4.41	name	The name attribute type is the attribute supertype from which string attribute types typically used for naming may be formed. It is unlikely that values of this type itself will occur in an entry. RFC 3280.
2.5.4.10	o	This attribute contains the name of an organization (organizationName). RFC 3280.
2.5.4.11	ou	This attribute contains the name of an organizational unit (organizationalUnitName). RFC 3280
2.5.4.17	postalCode	This attribute type specifies the postal code of the object. If the attribute value is present it will be part of the object's postal address.
2.5.4.65	pseudonym	According to RFC3039: ″pseudonym from(forthcoming) X.520″. RFC 3280.

OID	Naming Attribute	Description
2.5.4.5	serialNumber	This attribute contains the serial number of a device. RFC 3280.
2.5.4.4	sn	This is the X.500 surname attribute, which contains the family name of a person. RFC 3280.
2.5.4.8	st	This attribute contains the full name of a state or province (stateOrProvinceName). RFC 3280.
2.5.4.9	street	This attribute contains the physical address of the object to which the entry corresponds, such as an address for package delivery (streetAddress).
2.5.4.12	title	This attribute contains the title, such as Vice President, of a person in their organizational context. The personalTitle attribute would be used for a persons title independent of their job function. RFC 3280.
0.9.2342.19200300.100.1.1	uid	Typically a user shortname or userid.
2.5.4.45	x500UniqueIdentifier	Used to distinguish between objects when a distinguished name has been reused. This is a different attribute type from both the ″uid″ and ″uniqueIdentifier″ types.

Authorities and Locks

No authorization is required.

Parameters

userIdentityInfo (Input)

The user identity information from which to generate policy filter values.

The EimUserIdentityInfo structure contains information about the user identity.

For EIM_DER_CERT (0) or EIM_BASE64_CERT (1) user identity type, the *userIdentityInfo* field must contain an EimCertificate structure.

For EIM_CERT_INFO (2) user identity type, the *userIdentityInfo* field must contain an EimCertificateInfo structure.

The structure layouts follow:

```
enum EimUserIdentityType {
    EIM_DER_CERT,                  /* Entire X.509 public key
                                      certificate in ASN.1 DER
                                      encoding                    */
    EIM_BASE64_CERT,               /* Base 64 encoded version of the
                                      entire X.509 public key
                                      certificate in ASN.1 DER
                                      encoding.                   */
    EIM_CERT_INFO                  /* Components of the certificate. */
};

typedef struct EimCertificateInfo
{
    char          * issuerDN;      /* The issuer DN.              */
    char          * subjectDN;     /* The subject DN.             */
    unsigned char * publicKey;     /* The public key (may be NULL).  */
    unsigned int    publicKeyLen;  /* Length of public key (may be 0)*/
} EimCertificateInfo;

typedef struct EimCertificate
{
```

```
    char        * certData;         /* The certificate data           */
    unsigned int certLength;        /* The length of the certificate
                                       data.                          */
} EimCertificate;

typedef struct EimUserIdentityInfo
{
    enum EimUserIdentityType type;
    union {
        EimCertificateInfo certInfo;
        EimCertificate  cert;
    } userIdentityInfo;
} EimUserIdentityInfo;
```

If the *userIdentityInfo* field contains an EimCertificateInfo structure, the *issuerDN* and *subjectDN* fields must contain valid DN strings (for example, *CN=John D. Smith,OU=Sales,O=IBM,L=Rochester,ST=Min,C=US*). The *publicKey* field must contain the DER encoded public key information structure, including the tags and lengths.

subsetInfo (Input)

The information used to subset the policy filter values that are formatted. If NULL is specified for this parameter, then the returned data will contain all possible policy filter values for the specified user identity information. This option would be useful if you wanted to present the user with a list of possible policy filter values from which to choose. If this parameter is not NULL, then only one policy filter value will be returned based on the specified subset information.

The EimPolicyFilterSubsetInfo structure contains information for subsetting the return data. The information provided in the structure is dependent on the user identity type in the *userIdentityInfo* parameter.

For EIM_BASE64_CERT (0), EIM_DER_CERT (1), or EIM_CERT_INFO (2) user identity type, the *subset* field must contain an EimCertPolicyFilterSubsetInfo structure.

The structure layouts follow:

```
typedef struct EimCertPolicyFilterSubsetInfo
{
    char        * subjectFilter;    /* Subject filter value.          */
    char        * issuerFilter;     /* Issuer filter value.           */
} EimCertPolicyFilterSubsetInfo;

typedef struct EimPolicyFilterSubsetInfo
{
    union {
        EimCertPolicyFilterSubsetInfo   certFilter;
    } subset;
} EimPolicyFilterSubsetInfo;
```

lengthOfListData (Input)

The number of bytes provided by the caller for the *listData* parameter. 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 API will return as much data as space has been provided. The data returned is a linked list of EimPolicyFilterValue structures. firstEntry is used to get to the first EimPolicyFilterValue structure in the linked list.

EimList structure:

```
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;
```

EimPolicyFilterValue structure:

```
    typedef struct EimPolicyFilterValue
    {
        unsigned int nextEntry;         /* Displacement to next entry.  This
                                    byte offset is relative to the
                                    start of this structure          */
        EimListData filterValue;        /* Generated policy filter value. */
 } EimPolicyFilterValue;
```

EimListData structure:

```
    typedef struct EimListData
    {
        unsigned int length;            /* Length of data               */
        unsigned int disp;              /* Displacement to data.  This byte
                                    offset is relative to the start of
                                    the parent structure; that is, the
                                    structure containing this
                                    structure.                       */
    } EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EBADDATA
eimrc is not valid.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13)	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_PARM_REQ (34)	Missing required parameter. Please check API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.

EIMERR_SPACE (41)	Unexpected error accessing parameter.
EIMERR_USER_IDENTITY_TYPE_INVAL (63)	User identity type is not valid.
EIMERR_CERTIFICATE_INVAL (67)	Certificate data is not valid.

ENOMEM
Unable to allocate required space.

EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.

EUNKNOWN
Unexpected exception.

EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimAddPolicyFilter()—Add EIM Policy Filter" on page 31 —Add EIM Policy Filter
- "eimRemovePolicyFilter()—Remove EIM Policy Filter" on page 232 —Remove EIM Policy Filter
- "eimListPolicyFilters()—List EIM Policy Filters" on page 166 —List EIM Policy Filters

Example

See Code disclaimer information for information pertaining to code examples.

The following example generates certificate policy filter values.

```
#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[100];
    EimRC       * err;
    EimHandle   * handle;
    EimUserIdentityInfo  idInfo;
    char          listData[4000];
    EimList     * list = (EimList * ) listData;

    /* Set up error structure.                 */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get user identity information.          */
    idInfo.type = EIM_DER_CERT;
    idInfo.userIdentityInfo.cert.certLength = *((int *)argv[2]);
    idInfo.userIdentityInfo.cert.certData = argv[3];

    /* Format EIM Policy Filter                */
    /* This call will return all possible      */
```

```
    /* certificate policy filter values.        */
    if (0 != (rc = eimFormatPolicyFilter(&idInfo,
                                         NULL,
                                         4000,
                                         list,
                                         err)))
    {
        printf("Format EIM Policy Filter error = %d", rc);
        return -1;
    }


    /* Print the results                        */
    printListResults(list);

    return 0;

}

void printListResults(EimList * list)
{
    int i;
    EimPolicyFilterValue * entry;

    printf("___________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimPolicyFilterValue *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("===============\n");
        printf("Entry %d.\n", i);

        /* Print out results */
        printListData("Policy Filter Value",
                      entry,
                      offsetof(EimPolicyFilterValue, filterValue));

        /* advance to next entry */
        entry = (EimPolicyFilterValue *)((char *)entry + entry->nextEntry);

    }
    printf("\n");


}

void printListData(char * fieldName,
                   void * entry,
                   int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;
```

```
    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}
```

API introduced: V5R3

---

eimFormatUserIdentity()—Format User Identity

Syntax

```
#include <eim.h>

int eimFormatUserIdentity(
                     enum EimUserIdentityFormatType   formatType,
                          EimUserIdentityInfo       * userIdentityInfo,
                          unsigned int                lengthOfUserIdentity,
                          EimUserIdentity           * userIdentity,
                          EimRC                     * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimFormatUserIdentity()** function takes unformatted user identity information and formats it for use with other EIM functions.

Authorities and Locks

No authorization is required.

Parameters

formatType (Input)

How to format the user identity.

EIM_REGISTRY_USER_NAME (0)

Format the user identity into a registry user name. The registry user name will be normalized according to the normalization method for the *registryType*. This would be the registry user name that would be used as input to the Add EIM Association (eimAddAssociation) API. This data will be a NULL terminated string in the default CCSID of the job.

For certificates, the registry user name will be a combination of the subject DN, issuer DN, and a hash value of the subject DN, issuer DN, and public key. The registry user name will be in the format *<SDN>subject-DN</SDN><IDN>issuer-DN</IDN><HASH_VAL>hash-value</HASH_VAL>*.

userIdentityInfo (Input)

The user identity information to format.

The EimUserIdentityInfo structure contains information about the user identity to format.

For EIM_DER_CERT (0) or EIM_BASE64_CERT (1) user identity type, the *userIdentityInfo* field must contain an EimCertificate structure.

For EIM_CERT_INFO (2) user identity type, the *userIdentityInfo* field must contain an EimCertificateInfo structure.

The structure layouts follow:

```
enum EimUserIdentityType {
    EIM_DER_CERT,                  /* Entire X.509 public key
                                      certificate in ASN.1 DER
                                      encoding                    */
    EIM_BASE64_CERT,               /* Base 64 encoded version of the
                                      entire X.509 public key
                                      certificate in ASN.1 DER
                                      encoding.                   */
    EIM_CERT_INFO                  /* Components of the certificate. */
};

typedef struct EimCertificateInfo
{
    char          * issuerDN;      /* The issuer DN.              */
    char          * subjectDN;     /* The subject DN.             */
    unsigned char * publicKey;     /* The public key.             */
    unsigned int    publicKeyLen;  /* Length of the public key.   */
} EimCertificateInfo;

typedef struct EimCertificate
{
    unsigned int certLength;       /* The length of the certificate
                                      data.                       */
    char        * certData;        /* The certificate data        */
} EimCertificate;

typedef struct EimUserIdentityInfo
{
    enum EimUserIdentityType type;
    union {
        EimCertificateInfo certInfo;
        EimCertificate  cert;
    } userIdentityInfo;
} EimUserIdentityInfo;
```

If the *userIdentityInfo* field contains an EimCertificateInfo structure, the *issuerDN* and *subjectDN* fields must contain valid DN strings (for example, *CN=John D. Smith,OU=Sales,O=IBM,L=Rochester,ST=Min,C=US*). The *publicKey* field must contain the DER encoded public key information structure, including the tags and lengths.

NOTE: EIM recognizes all of the suggested naming attributes from RFC 3280 with a few additions. They are defined in the following table. If EIM encounters a naming attribute in a certificate that it does not recognize, the OID for the naming attribute will be used instead in the filter value. If you are using the eimCertificateInfo structure, the OID value for any naming attribute that is not in this table may be used.

OID	Naming Attribute	Description
2.5.4.6	c	This attribute contains a two-letter ISO 3166 country or region code (countryName). RFC 3280.
2.5.4.3	cn	This is the X.500 commonName attribute, which contains a name of an object. If the object corresponds to a person, it is typically the persons full name. RFC 3280.
0.9.2342.19200300.100.1.25	dc	Specifies one component of a domain name. RFC 3280.

OID	Naming Attribute	Description
2.5.4.46	dnQualifier	The dnQualifier attribute type specifies disambiguating information to add to the relative distinguished name of an entry. It is intended for use when merging data from multiple sources in order to prevent conflicts between entries which would otherwise have the same name. It is recommended that the value of the dnQualifier attribute be the same for all entries from a particular source. RFC3280.
1.2.840.113549.1.9.1	email	E-mail address
2.5.4.44	generationQualifier	Contains the part of the name which typically is the suffix, as in IIIrd. RFC 3280.
2.5.4.42	givenName	Used to hold the part of a persons name which is not their surname nor middle name. RFC 3280.
2.5.4.43	initials	The initials attribute contains the initials of some or all of an individuals names, but not the surname(s). RFC 3280.
2.5.4.7	l	This attribute contains the name of a locality, such as a city, county or other geographic region (localityName). RFC 3280.
0.9.2342.19200300.100.1.3	mail	Identifies a user's primary e-mail address (the e-mail address retrieved and displayed by "white-pages" lookup applications).
2.5.4.41	name	The name attribute type is the attribute supertype from which string attribute types typically used for naming may be formed. It is unlikely that values of this type itself will occur in an entry. RFC 3280.
2.5.4.10	o	This attribute contains the name of an organization (organizationName). RFC 3280.
2.5.4.11	ou	This attribute contains the name of an organizational unit (organizationalUnitName). RFC 3280
2.5.4.17	postalCode	This attribute type specifies the postal code of the object. If the attribute value is present it will be part of the object's postal address.
2.5.4.65	pseudonym	According to RFC3039: "pseudonym from (forthcoming) X.520". RFC 3280.
2.5.4.5	serialNumber	This attribute contains the serial number of a device. RFC 3280.
2.5.4.4	sn	This is the X.500 surname attribute, which contains the family name of a person. RFC 3280.
2.5.4.8	st	This attribute contains the full name of a state or province (stateOrProvinceName). RFC 3280.
2.5.4.9	street	This attribute contains the physical address of the object to which the entry corresponds, such as an address for package delivery (streetAddress).
2.5.4.12	title	This attribute contains the title, such as Vice President, of a person in their organizational context. The personalTitle attribute would be used for a persons title independent of their job function. RFC 3280.
0.9.2342.19200300.100.1.1	uid	Typically a user shortname or userid.

OID	Naming Attribute	Description
2.5.4.45	x500UniqueIdentifier	Used to distinguish between objects when a distinguished name has been reused. This is a different attribute type from both the “uid” and “uniqueIdentifier” types.

lengthOfUserIdentity (Input)
The number of bytes provided by the caller for the formatted user identify. Minimal size required is 16 bytes.

userIdentity (Output)
A pointer to the data to be returned.

The EimUserIdentity structure contains information about the returned data. The API will return as much data as space has been provided.

EimUserIdentity structure:

```
typedef struct EimUserIdentity
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API.                      */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API.                         */
    EimListData  userIdentity;      /* User identity                    */
} EimUserIdentity;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                   */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure; that is, the
                                       structure containing this
                                       structure.                       */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see “EimRC—EIM Return Code” on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EBADDATA
eimrc is not valid.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EINVAL
Input parameter was not valid.

EIMERR_PARM_REQ (34)	Missing required parameter. Please check API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_SPACE (41)	Unexpected error accessing parameter.
EIMERR_USER_IDENTITY_TYPE_INVAL (63)	User identity type is not valid.
EIMERR_USER_IDENTITY_SIZE (64)	Length of EimUserIdentity is not valid.
EIMERR_USER_IDENTITY_FORMAT_TYPE_INVAL (65)	User identity format type is not valid.
EIMERR_CERTIFICATE_INVAL (67)	Certificate data is not valid.

ENOMEM
Unable to allocate required space.

EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.

EUNKNOWN
Unexpected exception.

EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimAddAssociation()—Add EIM Association" on page 15 —Add EIM Association

Example

See Code disclaimer information for information pertaining to code examples.

The following example formats the user identity and adds an association.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;
    EimIdentifierInfo id;
    EimUserIdentityInfo  idInfo;
    char          rtnData[4000];
    EimUserIdentity   * fmtData = (EimUserIdentity * ) rtnData;

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get user identity information.           */
    idInfo.type = EIM_DER_CERT;
    idInfo.userIdentityInfo.cert.certLength = *((int *)argv[2]);
    idInfo.userIdentityInfo.cert.certData = argv[3];

    /* Format user identity                     */
    if (0 != (rc = eimFormatUserIdentity(EIM_REGISTRY_USER_NAME,
```

```
                                    &idInfo,
                                    4000,
                                    fmtData,
                                    err)))
    {
        printf("Format user identity error = %d", rc);
        return -1;
    }

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up identifier information            */
    id.idtype = EIM_UNIQUE_NAME;
    id.id.uniqueName = "mjones";

    /* Add the source association               */
    if (0 != (rc = eimAddAssociation(handle,
                                     EIM_SOURCE,
                                     &id,
                                     "MyX509Registry",
                                     (char *)fmtData + fmtData-&gtuserIdentity.disp,
                                     err)))
    {
        printf("Add Association error = %d", rc);
        return -1;
    }

    return 0;

}
```

API introduced: V5R3

Top | Security APIs | APIs by category

eimGetAssociatedIdentifiers() —Get Associated EIM identifiers

Syntax

```
#include <eim.h>

int eimGetAssociatedIdentifiers(EimHandle              * eim,
                                enum EimAssociationType  associationType,
                                char                   * registryName,
                                char                   * registryUserName,
                                unsigned int             lengthOfListData,
                                EimList                * listData,
                                EimRC                  * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimGetAssociatedIdentifiers()** function returns a list of the identifiers. Given a registry name and user name within that user registry, return the EIM identifier associated with it.

It is possible that more than one person is associated with a specific identifier. 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.

» If there are no specific associations for the registry name and user name within that registry to an EIM identifer, then group registries will be used. If the specified registry is a member of any group registries, then this API will return any EIM identifiers associated with the group registry (or registries) and the registry user name. «

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator
- EIM Mapping Lookup
- EIM authority to an individual registry

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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

associationType (Input)

The type of association to be retrieved. Valid values are:

EIM_ALL_ASSOC (0)	Retrieve all target, source, and administrative 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. A NULL parameter indicates that the `localRegistry` set by the eimSetConfiguration() API or the eimSetConfigurationExt() API should be used.

registryUserName (Input)

The registry user name for the lookup.

lengthOfListData (Input)

The number of bytes provided by the caller 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 API will return as much data as space has been provided. The data returned is a linked list of EimIdentifier structures. firstEntry is used to get to the first EimIdentifier structure in the linked list.

EimList structure:

```
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;
```

EimIdentifier structure:

```
typedef struct EimIdentifier
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                    byte offset is relative to the
                                    start of this structure          */
    EimListData uniquename;         /* Unique name                      */
    EimListData description;        /* Description                      */
    EimListData entryUUID;          /* UUID                             */
    EimSubList  names;              /* EimIdentifierName sublist        */
    EimSubList  additionalInfo;     /* EimAddlInfo sublist              */
    enum EimAssociationType type;   /* Association type                 */

    EimListData groupRegistry;      /* Group registry used to get the
                                    identifier.                      */

} EimIdentifier;
```

Identifiers may have several name attributes as well as several additional information attributes. In the EimIdentity structure, the names EimSubList gives addressability to a linked list of EimIdentifierName structures.

EimIdentifierName structure:

```
 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.

EimAddlInfo structure:

```
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;
```

EimSubList structure:

```
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;
                                    that is, the structure containing
                                    this structure.                  */
} EimSubList;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;           /* Length of data                */
    unsigned int disp;             /* Displacement to data.  This byte
                                    offset is relative to the start of
                                    the parent structure; that is, the
                                    structure containing this
                                    structure.                       */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry not found or insufficient access to EIM data.

EIMERR_NOREG (28) EIM Registry not found or insufficient access to EIM data.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35)	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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23)	Unexpected LDAP error. %s
EIMERR_UNEXP_OBJ_ VIOLATION (56)	Unexpected object violation.
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimAddIdentifier()—Add EIM Identifier" on page 23—Add EIM Identifier
- "eimChangeIdentifier()— Change EIM Identifier" on page 41—Change EIM Identifier
- "eimRemoveIdentifier()— Remove EIM Identifier" on page 224—Remove EIM Identifier
- "eimListIdentifiers()— List EIM Identifiers" on page 160—List EIM Identifiers

Example

See Code disclaimer information for information pertaining to code examples.

The following example will list all of the identifiers 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);
void printAssociationType(int type);

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
```

```
    EimRC       * err;
    EimHandle   * handle;

    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get associated identifiers               */
    if (0 != (rc = eimGetAssociatedIdentifiers(handle,
                                           EIM_ALL_ASSOC,
                                           "MyRegistry",
                                           "carolb",
                                           1000,
                                           list,
                                           err)))
    {
        printf("Get Associated Identifers error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    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));
        printAssociationType(entry->type);
        printListData("Group registry",
                      entry,
                      offsetof(EimIdentifier, groupRegistry));

        /* 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");

}

void printAssociationType(int type)
{
    switch(type)
    {
        case EIM_SOURCE:
            printf("    Source Association.\n");
            break;
        case EIM_ADMIN:
            printf("    Admin Association.\n");
            break;
        case EIM_TARGET:
            printf("    Target Association.\n");
            break;
        default:
            printf("ERROR - unknown association type.\n");
            break;
    }
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimGetAttribute()—Get EIM attributes

Syntax

```
#include <eim.h>

int eimGetAttribute(EimHandle          * eim,
                    enum EimHandleAttr   attrName,
                    unsigned int         lengthOfEimAttribute,
                    EimAttribute       * attribute,
        EimRC              * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimGetAttribute()** function is used to get attributes for this EIM handle.

The ldap configuration file is used to retrieve information for the master host, master port, and secure port. 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.

Authorities and Locks

None.

Parameters

eim (Input)
The EIM handle returned by a previous call to eimCreateHandle().

attrName (Input)
The name of the attribute to retrieve. Following are valid values:

EIM_HANDLE_CCSID (0)	This is the CCSID of character data passed by the caller of EIM APIs 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: • 0 - Non-SLL • 1- Port uses SSL
EIM_HANDLE_MASTER_HOST (5)	If the EIM_HANDLE_HOST is a replica LDAP server, this value will indicate the master LDAP server.
EIM_HANDLE_MASTER_PORT (6)	If the EIM_HANDLE_HOST is a replica LDAP server, this value will indicate 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 will indicate the security type for the master LDAP server. The returned field is a 4 byte integer.

lengthOfEimAttribute (Input)
The number of bytes provided by the caller for the attribute information. Minimum size required is 16 bytes.

attribute (Output)
A pointer to the data to be returned.

The EimAttribute structure contains information about the returned data. The API will return as much data as space has been provided.

EimAttribute structure:

```
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;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                   */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure; that is, the
                                       structure containing this
                                       structure.                       */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35) 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. When configured for SSL we cannot retrieve the master information until a connection has been established to the configured system.

EIMERR_NOT_CONN (31) Not connected to LDAP. Use eimConnect() API and try the request again.

ENOTSUP
Attribute type is not supported.

EIMERR_ATTR_NOTSUPP (6) Attribute not supported.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23)	Unexpected LDAP error. %s
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimCreateHandle()—Create an EIM Handle" on page 70—Create an EIM Handle
- "eimDestroyHandle()—Destroy an EIM Handle" on page 76—Destroy an EIM Handle
- "eimSetAttribute()—Set EIM attributes" on page 239—Set EIM Attributes
- "eimConnectToMaster()—Connect to EIM Master Domain" on page 60—Connect to EIM Master Domain
- "eimConnect()—Connect to EIM Domain" on page 57—Connect to EIM Domain

Example

See Code disclaimer information for information pertaining to code examples.

The following example will get the domain for the EIM handle.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int            rc;
    char           eimerr[100];
    EimRC        * err;
    EimHandle    * handle;
    char         * data;
    char         * listData[1000];
    EimAttribute * list = (EimAttribute *) listData;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get EIM domain name                      */
    if (0 != (rc = eimGetAttribute(handle,
                                   EIM_HANDLE_DOMAIN,
                                   1000,
                                   list,
                                   err)))
        printf("Get Attribute error = %d", rc);

    /* Print results                            */
    printf("   Bytes returned  = %d.\n", list->bytesReturned);
    printf("   Bytes available = %d.\n", list->bytesAvailable);

    printf("   Attr size = %d.\n", list->attribute.length);
    printf("   Attr disp = %d.\n", list->attribute.disp);

    data = (char * )list + list->attribute.disp;
```

```
    printf("     %s = %s.\n", "domain name", data);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

QsyGetEIMConnectInfo()—Get EIM Connect Information

Syntax

```
#include <qsyeimapi.h>

#include <eim.h>

int QsyGetEIMConnectInfo(int          lengthOfConnectInfo,
    EimList    * connectInfo,
                         EimRc      * eimrc)
```

Service Program Name: QSYS/QSYEIMAPI
Default Public Authority: *USE
Threadsafe: Yes

The **QsyGetEIMConnectInfo()** function returns the connection information that will be used by the operating system when it needs to connect to the EIM domain that is configured for this system or for the master system.

Authorities and Locks

None.

Parameters

lengthOfConnectInfo

(Input)

The number of bytes provided by the caller for the connection information parameter. The minimum size required is 20 bytes. The API will return the number of bytes available for all of the connection information and as much data as space has been provided.

connectInfo

(Output)

A pointer to the data to be returned.

The EimList structure contains information about the returned data. The data returned is a linked list of QsyEimConnectInfo structures. firstEntry is used to get to the first QsyEimConnectInfo structure in the linked list.

EimList structure:

```
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;
```

QsyEimConnectInfo structure:

```
#pragma enumsize(4)

typedef struct QsyEimConnectInfo
{
     unsigned int              nextEntry;
                                    /* Displacement to next entry.
                                       This byte offset is relative
                                       to the start of this structure.  */
     enum QsyEimConnectSystem  connectSystem;
                                    /* System connection info is for -
                                       configured (0) or master (1).    */
     enum QsyEimConnectType    connectType;
                                    /* Connection type - simple (0),
                                       kerberos with keytab file (1), or
                                       kerberos with password (2)       */
     union {
         struct {
             enum EimPasswordProtect  protect;
                                    /* Protect value - no protect (0),
                                       cram_md5 (1), or optional
                                       cram_md5 (2)                     */
             EimListData              bindDN;
         } simpleConnect;           /* Protect value and bind DN, if
                                       connectType=QSY_EIM_SIMPLE (0)   */
         struct {
             EimListData       kerberosPrincipal;
             EimListData       kerberosRealm;
         } kerberosPwd;             /* Kerberos information, if
                                       connectType=QSY_KERBEROS_PWD (1) */
         struct {
             EimListData       kerberosKeyTab;
             EimListData       kerberosPrincipal;
             EimListData       kerberosRealm;
         } kerberosKeyTab;          /* Kerberos information, if
                                       connectType=
                                         QSY_KERBEROS_KEYTAB (2)        */

     } connectInfo;
} QsyEimConnectInfo;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                  */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure; that is, the
                                       structure containing this
                                       structure                       */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the messageCatalogMessageID field in the eimrc parameter for that value.

0 Request was successful.

EBADDATA (3028)
eimrc is not valid.

EBUSY (3029)
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT (3490)
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EINVAL (3021)
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_PARM_REQ (34) Missing required parameter. Please check API documentation.
EIMERR_PTR_INVAL (35) Pointer parameter is not valid.
EIMERR_SPACE (41) Unexpected error accessing parameter.

ENOMEM (3460)
Unable to allocate required space.

EIMERR_NOMEM (27) No memory available. Unable to allocate required space.

EUNKNOWN (3474)
Unexpected exception.

EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "QsySetEIMConnectInfo()—Set EIM Connect Information" on page 250—Set EIM Connect Information

Example

See Code disclaimer information for information pertaining to code examples.

The following example will get connection information used by the operating system.

```
#include <eim.h>
#include <qsyeimapi.h>

void printListResults(EimList * list)
```

```
{
    int i;
    QsyEimConnectInfo * entry;
    char * data;
    int    dataLength;

    printf("\n__________");
    printf("\nBytes Returned    = %d", list->bytesReturned);
    printf("\nBytes Available   = %d", list->bytesAvailable);
    printf("\nEntries Returned  = %d", list->entriesReturned);
    printf("\nEntries Available = %d", list->entriesAvailable);

    entry = (QsyEimConnectInfo *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("\n***** Entry %d *****  ",i+1);
        printf("\nConnect system : %d ",entry->connectSystem );
        printf("\nConnect type : %d ",entry->connectType );

        switch (entry->connectType)  /* Determine connect type. */
          {
            case QSY_EIM_SIMPLE:
               {
                 printf("\nProtect type : %d ",
                   entry->connectInfo.simpleConnect.protect );
                 data = ((char *)entry +
                   entry->connectInfo.simpleConnect.bindDN.disp );
                 dataLength =
                   entry->connectInfo.simpleConnect.bindDN.length;
                 printf("\n%s : ","Bind DN");
                 if (dataLength > 0)
                     printf("%.*s",dataLength, data);
                 else
                     printf("Not found.");
                 break;
               }
            case QSY_EIM_KERBEROS_KEYTAB:
               {
                 /* Print out the keytab file name */
                 data = ((char *)entry + entry->
                      connectInfo.kerberosKeyTab.kerberosKeyTab.disp );
                 dataLength =
                   entry->connectInfo.kerberosKeyTab.kerberosKeyTab.length;
                 printf("\n%s : ","Keytab file name");
                 if (dataLength > 0)
                     printf("%.*s",dataLength, data);
                 else
                     printf("Not found.");
                 /* Print out the principal */
                 data = ((char *)entry + entry->
                       connectInfo.kerberosKeyTab.kerberosPrincipal.disp );
                 dataLength =
                   entry->connectInfo.kerberosKeyTab.kerberosPrincipal.length;
                 printf("\n%s : ","Kerberos principal");
                 if (dataLength > 0)
                     printf("%.*s",dataLength, data);
                 else
                     printf("Not found.");
                 /* Print out the realm */
                 data = ((char *)entry + entry->
                      connectInfo.kerberosKeyTab.kerberosRealm.disp );
                 dataLength =
                   entry->connectInfo.kerberosKeyTab.kerberosRealm.length;
                 printf("\n%s : ","Kerberos realm");
                 if (dataLength > 0)
                     printf("%.*s",dataLength, data);
```

```
                else
                    printf("Not found.");
                break;
              }
            case QSY_EIM_KERBEROS_PWD:
              {
                /* Print out the principal */
                data = ((char *)entry + entry->
                      connectInfo.kerberosPwd.kerberosPrincipal.disp );
                dataLength =
                  entry->connectInfo.kerberosPwd.kerberosPrincipal.length;
                printf("\n%s : ","Kerberos principal");
                if (dataLength > 0)
                    printf("%.*s",dataLength, data);
                else
                    printf("Not found.");
                /* Print out the realm */
                data = ((char *)entry + entry->
                      connectInfo.kerberosPwd.kerberosRealm.disp );
                dataLength =
                  entry->connectInfo.kerberosPwd.kerberosRealm.length;
                printf("\n%s : ","Kerberos realm");
                if (dataLength > 0)
                    printf("%.*s",dataLength, data);
                else
                    printf("Not found.");
                break;
              }
          } /* end determine connect type. */

        /* advance to next entry */
        entry = (QsyEimConnectInfo *)((char *)entry + entry->nextEntry);

    }
    printf("\n");

}

int main(int argc, char *argv[])
{
    int rc;
    char       eimerr[100];
    EimRC     *err;

    char    listData[5000];
    EimList * list = (EimList * ) listData;

    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    if (0 != (rc = QsyGetEIMConnectInfo(5000,
                                        list,
                                        err)))
    {
        printf("Get connection information error = %d", rc);
        return -1;
    }

    printListResults(list);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

QsyGetEIMHandle()—Get EIM Handle Connected For System

Syntax

```
#include <qsyeimapi.h>

#include <eim.h>

int QsyGetEIMHandle(EimHandle      * eim,,
                    EimRc          * eimrc)
```

Service Program Name: QSYS/QSYEIMAPI
Default Public Authority: *USE
Threadsafe: Yes

The **QsyGetEIMHandle()** function is used to allocate an EimHandle structure that is connected to EIM. The EIM host information used will be the `ldapURL` information set by the eimSetConfiguration() API. The connection information used will be the `connectInfo` set by the QsySetEIMConnectInfo() API. The EimHandle structure should be passed on subsequent calls to other EIM operations. When the handle is no longer needed, it should be destroyed by calling **eimDestroyHandle()** function.

Authorities and Locks

Authority required
*ALLOBJ and *SECADM special authorities

Parameters

eim (Output)
The pointer to an EIM handle to be returned. This handle is used as input for other EIM APIs. The handle is temporary; you can use it only in the job that created it.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the messageCatalogMessageID field in the eimrc parameter for that value.

0 Request was successful.

EACCES
Access denied.

EIMERR_AUTH_ERR (7) Insufficient authority for the operation.

EBADDATA (3028)
eimrc is not valid.

EBUSY (3029)
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT (3490)
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EINVAL (3021)
Input parameter was not valid.

EIMERR_SSL_REQ (42)	The system is configured to connect to a secure port. EimSSLInfo is required.
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_URL_NOTLDAP (49)	URL does not begin with ldap.
EIMERR_URL_NOHOST (47)	URL does not have a host.

ENOMEM (3460)
Unable to allocate required space.

EIMERR_NOMEM (27) No memory available. Unable to allocate required space.

ENOSYS (3470)
System not configured.

EIMERR_OS400_NOTSET_CONFIG (5007)	EIM connection information is not set for this system. Run QsySetEIMConnectInfo() API and try the request again.
EIMERR_OS400_NOTSET_MASTER (5008)	EIM connection information is not set for the master system. Run QsySetEIMConnectInfo() API and try the request again.

EUNKNOWN (3474)
Unexpected exception.

EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "QsySetEIMConnectInfo()—Set EIM Connect Information" on page 250—Get EIM Connect Information
- "QsySetEIMConnectInfo()—Set EIM Connect Information" on page 250—Set EIM Connect Information
- "eimDestroyHandle()—Destroy an EIM Handle" on page 76—Destroy an EIM Handle

Example

See Code disclaimer information for information pertaining to code examples.

The following example will get connection information used by the operating system.

```
#include <eim.h>
#include <qsyeimapi.h>


int main(int argc, char *argv[])
{
    int rc;
    char       eimerr[100];
    EimRC     *err;
    EimHandle     handle;

    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    if (0 != (rc = QsyGetEIMHandle(&handle,
                                       err)))
    {
        printf("Get connected handle error = %d", rc);
        return -1;
    }

    ...  other eim operations


    if (0 != (rc = eimDestroyHandle(&handle,
                                       err)))
    {
        printf("Destroy eim handle error = %d", rc);
        return -1;
    }


    return 0;
}
```

API introduced: V5R4

Top | Security APIs | APIs by category

---

eimGetRegistryNameFromAlias() —Get EIM Registry Name from an Alias

Syntax

```
#include <eim.h>

int eimGetRegistryNameFromAlias(EimHandle      * eim,
                                char           * aliasType,
                                char           * aliasValue,
                                unsigned int     lengthOfListData,
                                EimList        * listData,
                                EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimGetRegistryNameFromAlias()** function will return a list of registry names that match the search criteria provided by *aliasType* and *aliasValue*.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator
- EIM Mapping Lookup
- EIM authority to an individual registry

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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

aliasType (Input)

The type of alias for which to search. See eim.h for a list of predefined alias types.

aliasValue (Input)

The value for this alias.

lengthOfListData (Input)

The number of bytes provided by the caller 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 API will return as much data as space has been provided. The data returned is a linked list of EimRegistryName structures. firstEntry is used to get to the first EimRegistryName structure in the linked list.

EimList structure:

```
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;
```

EimRegistryName structure:

```
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;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;           /* Length of data                */
    unsigned int disp;             /* Displacement to data.  This byte
                                    offset is relative to the start of
                                    the parent structure; that is, the
                                    structure containing this
                                    structure.                       */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35) 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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimChangeRegistryAlias()—Change EIM Registry Alias" on page 49 —Change EIM Registry Alias
- "eimListRegistryAliases()—List EIM Registry Aliases" on page 179 —List EIM Registry Aliases

Example

See Code disclaimer information for information pertaining to code examples.

The following example will get 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[100];
    EimRC       * err;
    EimHandle   * handle;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get registry from alias                  */
    if (0 != (rc = eimGetRegistryNameFromAlias(handle,
                                               EIM_ALIASTYPE_DNS,
                                               "Clueless",
                                               1000,
                                               list,
```

```
                                            err)))
    {
        printf("Get registry name from alias error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    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");

}
```

API introduced: V5R2

eimGetTargetCredsFromSource() —Get EIM Target Identities and Credentials from the Source

Syntax

```
#include <eim.h>

int eimGetTargetCredsFromSource(EimHandle      * eim,
                                char           * sourceRegistryName,
                                char           * sourceRegistryUserName,
                                char           * targetRegistryName,
                                char           * additionalInformation,
                                unsigned int     lengthOfListData,
                                EimList        * listData,
                                EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimGetTargetCredsFromSource()** function gets the target identity(ies) and credentials 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 return the user for this target registry.

EIM version 3 must be supported by the local EIM APIs to use this API (see “eimGetVersion()—Get EIM Version” on page 140—Get EIM Version).

See EIM Mapping Lookup Algorithm for the steps involved in a mapping lookup operation.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the mapping lookup data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator
- EIM Mapping Lookup
- EIM authority to an individual registry

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

The credential information for the target identity is considered security sensitive data. Access to this data is more strictly controlled. The access groups whose members have authority to the credential information for the target identity follow:

- EIM Administrator
- EIM Credential Data
- EIM authority to an individual registry

Note that the EIM Credential Data access group does not have access to the mapping lookup data. If a user is a member of the EIM Credential Data access group, then the user must also be a member of one of the access groups that has access to the mapping lookup data.

Parameters

eim (Input)
The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

sourceRegistryName (Input)
The source registry for this lookup operation.

sourceRegistryUserName (Input)
The source user name for this lookup operation.

targetRegistryName (Input)
The target registry for this lookup operation. A NULL parameter indicates that the `localRegistry` set by the eimSetConfiguration() API or the eimSetConfigurationExt() API should be used.

additionalInfo (Input)
Additional information that will be used as selection criteria for this operation. This may be NULL. This filter data may contain the wild card char(*).

lengthOfListData (Input)
The number of bytes provided by the caller 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 API will return as much data as space has been provided. The data returned is a linked list of EimTargetIdentity structures. firstEntry is used to get to the first EimTargetIdentity structure in the linked list. Each EimTargetIdentity entry contains a user name returned by this lookup operation.

EimList structure:

```
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;
```

EimTargetIdentity structure:

```
typedef struct EimTargetIdentity
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure          */
    EimListData userName;           /* User name                        */
    enum EimAssociationType type;   /* Association type                 */
```

```
    EimListData sourceGroupRegistry;/* Source group registry name     */
    EimListData targetGroupRegistry;/* Target group registry name     */
    EimSubList  credentialInfo;     /* EimCredentialInfo sublist      */
} EimTargetIdentity;
```

The *sourceGroupRegistry* will be returned if the target identity was found using a source association to a group registry. The *targetGroupRegistry* will be returned if the target identity was found using a target association to a group registry.

Target identities may have several types of credentials. In the EimTargetIdentity structure, credentialInfo gives addressability to the first EimCredentialInfo structure that contains a linked list of credentials.

If there is credential information for the target identity, but the caller is not authorized to access the credential information or the credential data is not enabled, the EimCredentialInfo structure will be returned with the *type* and *status* fields filled in. The *data* field will not be returned (*length* and *disp* will be 0). If there is no credential information, the EimCredentialInfo structure will not be returned in the *credentialInfo* sublist.

EimCredentialInfo structure:

```
typedef struct EimCredentialInfo
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure.         */
    enum EimCredentialType type;    /* Credential type                  */
    enum EimStatus         status;  /* Credential status
                                       0 = not enabled
                                       1 = enabled                      */
    EimListData            data;    /* Credential data                  */
} EimCredentialInfo;
```

EimSubList structure:

```
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;
                                       that is, the structure containing
                                       this structure.                  */
} EimSubList;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                 */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure; that is, the
                                       structure containing this
                                       structure.                       */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry not found or insufficient access to EIM data.

EIMERR_NOREG (28) EIM Registry not found or insufficient access to EIM data.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_SPACE (41)	Unexpected error accessing parameter.
EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified function is not supported by the EIM version.

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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s
EIMERR_UNEXP_OBJ_VIOLATION (56) Unexpected object violation.
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimGetTgtCredsFromIdentifier() —Get EIM Target Identities and Credentials from the Identifier" on page 121 —Get EIM Target Identities and Credentials from the Identifier

Example

See Code disclaimer information for information pertaining to code examples.

The following example will get the list of users and credentials in the target registry, MyRegistry, that are 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);
void printAssociationType(int type);
void printCredSubListData(char * fieldName,
                          void * entry,
                          int offset);

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get target identity                      */
```

```
    if (0 != (rc = eimGetTargetCredsFromSource(handle,
                                               "kerberosRegistry",
                                               "mjjones",
                                               "MyRegistry",
                                               NULL,
                                               1000,
                                               list,
                                               err)))
    {
        printf("Get target credentials from source error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    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));
        printAssociationType(entry->type);
        printListData("source group registry",
                      entry,
                      offsetof(EimTargetIdentity, sourceGroupRegistry));
        printListData("target group registry",
                      entry,
                      offsetof(EimTargetIdentity, targetGroupRegistry));
        printCredSubListData("credential information",
                             entry,
                             offsetof(EimTargetIdentity, credentialInfo));

        /* advance to next entry */
        entry = (EimTargetIdentity *)((char *)entry + entry->nextEntry);

    }
    printf("\n");


}

void printListData(char * fieldName,
                   void * entry,
                   int offset)
{
```

```
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}

void printAssociationType(int type)
{
    switch(type)
    {
        case EIM_TARGET:
            printf("    Target Association.\n");
            break;
        case EIM_CERT_FILTER_POLICY:
            printf("    Certificate Filter Policy Association.\n");
            break;
        case EIM_DEFAULT_REG_POLICY:
            printf("    Default Registry Policy Association.\n");
            break;
        case EIM_DEFAULT_DOMAIN_POLICY:
            printf("    Default Domain Policy Association.\n");
            break;
        default:
            printf("ERROR - unknown association type.\n");
            break;
    }
}

void printCredSubListData(char * fieldName,
                          void * entry,
                          int offset)
{
    int i;
    EimSubList * subList;
    EimCredentialInfo * subentry;

    /* Address the EimSubList object */
    subList = (EimSubList *)((char *)entry + offset);

    if (subList->listNum > 0)
    {
        subentry = (EimCredentialInfo *)((char *)entry + subList->disp);
        for (i = 0; i < subList->listNum; i++)
        {

            /* Print out results */
            printf("     Credential type = %d.\n",subentry->type);
            printf("     Credential status = %d.\n",subentry->status);
            /* Credential data is not printed.                */

            /* advance to next entry */
            subentry = (EimCredentialInfo *)((char *)subentry +
                                             subentry->nextEntry);
```

```
        }
    }
}
```

« API introduced: V5R4

eimGetTgtCredsFromIdentifier() —Get EIM Target Identities and Credentials from the Identifier

Syntax

```
#include <eim.h>

int eimGetTgtCredsFromIdentifier(EimHandle         * eim,
                                 EimIdentifierInfo * idName,
                                 char              * targetRegistryName,
                                 char              * additionalInformation,
                                 unsigned int        lengthOfListData,
                                 EimList           * listData,
                                 EimRC             * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimGetTgtCredsFromIdentifier()** function gets the target identity or identities and credentials for the specified registry that is associated with the specified EIM identifier.

EIM version 3 must be supported by the local EIM APIs to use this API (see "eimGetVersion()—Get EIM Version" on page 140—Get EIM Version).

See EIM Mapping Lookup Algorithm for the steps involved in a mapping lookup operation.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the mapping lookup data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator
- EIM Mapping Lookup
- EIM authority to an individual registry

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

The credential information for the target identity is considered security sensitive data. Access to this data is more strictly controlled. The access groups whose members have authority to the credential information for the target identity follow:

- EIM Administrator
- EIM Credential Data
- EIM authority to an individual registry

Note that the EIM Credential Data access group does not have access to the mapping lookup data. If a user is a member of the EIM Credential Data access group, then the user must also be a member of one of the access groups that has access to the mapping lookup data.

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

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` indicates which identifier name is provided. Use of the `uniqueName` provides the best performance. Specifying an `idtype` of EIM_NAME does not guarantee that a unique EIM identifier will be found. Therefore, use of EIM_NAME may result in an error.

targetRegistryName (Input)

The target registry for this lookup operation. A NULL parameter indicates that the `localRegistry` set by the eimSetConfiguration() API or the eimSetConfigurationExt() API should be used.

additionalInfo (Input)

Additional information that will be used as selection criteria for this operation. This may be NULL.

lengthOfListData (Input)

The number of bytes provided by the caller 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 API will return as much data as space has been provided. The data returned is a linked list of EimTargetIdentity structures. firstEntry is used to get to the first EimTargetIdentity structure in the linked list. Each EimTargetIdentity entry contains a user name returned by this lookup operation.

EimList structure:

```
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;
```

EimTargetIdentity structure:

```
typedef struct EimTargetIdentity
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure.        */
    EimListData userName;           /* User name                       */
    enum EimAssociationType type;   /* Association type                */
    EimListData sourceGroupRegistry;/* Source group registry name      */
    EimListData targetGroupRegistry;/* Target group registry name      */
    EimSubList  credentialInfo;     /* EimCredentialInfo sublist       */
} EimTargetIdentity;
```

The *sourceGroupRegistry* will not be returned by this API. The *targetGroupRegistry* will be returned if the target identity was found using a target association to a group registry.

Target identities may have several types of credentials. In the EimTargetIdentity structure, credentialInfo gives addressability to the first EimCredentialInfo structure that contains a linked list of credentials.

If there is credential information for the target identity, but the caller is not authorized to access the credential information or the credential data is not enabled, the EimCredentialInfo structure will be returned with the *type* and *status* fields filled in. The *data* field will not be returned (*length* and *disp* will be 0). If there is no credential information, the EimCredentialInfo structure will not be returned in the *credentialInfo* sublist.

EimCredentialInfo structure:

```
typedef struct EimCredentialInfo
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure.        */
    enum EimCredentialType type;    /* Credential type                 */
    enum EimStatus         status;  /* Credential status
                                         0 = not enabled
                                         1 = enabled                   */
    EimListData            data;    /* Credential data                 */
} EimCredentialInfo;
```

EimSubList structure:

```
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;
                                       that is, the structure containing
                                       this structure.                 */
} EimSubList;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;           /* Length of data                */
    unsigned int disp;             /* Displacement to data.  This byte
                                    offset is relative to the start of
                                    the parent structure; that is, the
                                    structure containing this
                                    structure.                       */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry or identifier not found or insufficient access to EIM data.

EIMERR_IDNAME_AMBIGUOUS (20) More than 1 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) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_SPACE (41)	Unexpected error accessing parameter.
EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified function is not supported by the EIM version.

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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23)	Unexpected LDAP error. %s
EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimGetTargetCredsFromSource() —Get EIM Target Identities and Credentials from the Source" on page 114 —Get EIM Target Identities and Credentials from the Source

Example

See Code disclaimer information for information pertaining to code examples.

The following example will get the list of users and credentials in the target registry, MyRegistry, that are 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);
void printAssociationType(int type);
void printCredSubListData(char * fieldName,
                          void * entry,
```

```
                        int offset);


int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    EimIdentifierInfo x;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Set up identifier information */
    x.idtype = EIM_UNIQUE_NAME;
    x.id.uniqueName = "mjones";

    if (0 != (rc = eimGetTgtCredsFromIdentifier(handle,
                                                &x,
                                                "MyRegistry",
                                                NULL,
                                                1000,
                                                list,
                                                err)))
    {
        printf("Get target credentials from identifier error = %d", rc);
        return -1;
    }

    printListResults(list);

    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));
        printAssociationType(entry->type);
        printListData("target group registry",
                      entry,
                      offsetof(EimTargetIdentity, targetGroupRegistry));
        printCredSubListData("credential information",
                             entry,
                             offsetof(EimTargetIdentity, credentialInfo));

        /* advance to next entry */
        entry = (EimTargetIdentity *)((char *)entry + entry->nextEntry);

    }
    printf("\n");


}

void printListData(char * fieldName,
                   void * entry,
                   int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}

void printAssociationType(int type)
{
    switch(type)
    {
        case EIM_TARGET:
            printf("    Target Association.\n");
            break;
        case EIM_DEFAULT_DOMAIN_POLICY:
            printf("    Default Domain Policy Association.\n");
            break;
        default:
            printf("ERROR - unknown association type.\n");
            break;
    }
}

void printCredSubListData(char * fieldName,
                          void * entry,
                          int offset)
{
    int i;
    EimSubList * subList;
    EimCredentialInfo * subentry;
```

```
    /* Address the EimSubList object */
    subList = (EimSubList *)((char *)entry + offset);

    if (subList->listNum > 0)
    {
        subentry = (EimCredentialInfo *)((char *)entry + subList->disp);
        for (i = 0; i < subList->listNum; i++)
        {

            /* Print out results */
            printf("     Credential type = %d.\n",subentry->type);
            printf("     Credential status = %d.\n",subentry->status);
            /* Credential data is not printed.               */

            /* advance to next entry */
            subentry = (EimCredentialInfo *)((char *)subentry +
                                        subentry->nextEntry);
        }
    }

}
```

« API introduced: V5R4

Top | Security APIs | APIs by category

---

eimGetTargetFromIdentifier() —Get EIM Target Identities from the Identifier

Syntax

```
#include <eim.h>

int eimGetTargetFromIdentifier(EimHandle         * eim,
                               EimIdentifierInfo * idName,
                               char              * targetRegistryName,
                               char              * additionalInformation,
                               unsigned int        lengthOfListData,
                               EimList           * listData,
                               EimRC             * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimGetTargetFromIdentifier()** function gets the target identity or identities for the specified registry that is associated with the specified EIM identifier.

» See EIM Mapping Lookup Algorithm for the steps involved in a mapping lookup operation. «

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator
- EIM Mapping Lookup

- EIM authority to an individual registry

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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

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` indicates which identifier name is provided. Use of the `uniqueName` provides the best performance. Specifying an `idtype` of EIM_NAME does not guarantee that a unique EIM identifier will be found. Therefore, use of EIM_NAME may result in an error.

targetRegistryName (Input)

The target registry for this lookup operation. A NULL parameter indicates that the `localRegistry` set by the eimSetConfiguration() API or the eimSetConfigurationExt() API should be used.

additionalInfo (Input)

Additional information that will be used as selection criteria for this operation. This may be NULL.

lengthOfListData (Input)

The number of bytes provided by the caller 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 API will return as much data as space has been provided. The data returned is a linked list of EimTargetIdentity structures. firstEntry is used to get to the first EimTargetIdentity structure in the linked list. Each EimTargetIdentity entry contains a user name returned by this lookup operation.

EimList structure:

```
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;
```

EimTargetIdentity structure:

```
typedef struct EimTargetIdentity
{
    unsigned int nextEntry;        /* Displacement to next entry.  This
                                    byte offset is relative to the
                                    start of this structure.        */
    EimListData userName;          /* User name                       */
    enum EimAssociationType type;  /* Association type                */

    EimListData sourceGroupRegistry;/* Source group registry name     */
    EimListData targetGroupRegistry;/* Target group registry name     */
    EimSubList  credentialInfo;    /* EimCredentialInfo sublist       */

} EimTargetIdentity;
```

» The *sourceGroupRegistry* will not be returned by this API. The *targetGroupRegistry* will be returned if the target identity was found using a target association to a group registry.

This API will always return 0 for the numbers of entries in the credentialInfo sublist. If you need access to the credential information, use the Get EIM Target Identities and Credentials from the Identifier (eimGetTgtCredsFromIdentifier) API.

EimSubList structure:

```
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;
                                    that is, the structure containing
                                    this structure.                 */
} EimSubList;
```

«

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;           /* Length of data                  */
    unsigned int disp;             /* Displacement to data.  This byte
                                    offset is relative to the start of
                                    the parent structure; that is, the
                                    structure containing this
                                    structure.                      */
} EimListData;
```

eimrc **(Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1)	Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry or identifier not found or insufficient access to EIM data.

EIMERR_IDNAME_AMBIGUOUS (20)	More than 1 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)	Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13)	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 API documentation.
EIMERR_PTR_INVAL (35)	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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s
EIMERR_UNEXP_OBJ_VIOLATION (56) Unexpected object violation.
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimGetTargetFromSource() —Get EIM Target Identities from the Source" on page 134 —Get EIM Target Identities from the Source
- "eimGetTgtCredsFromIdentifier() —Get EIM Target Identities and Credentials from the Identifier" on page 121 —Get EIM Target Identities and Credentials from the Identifier

Example

See Code disclaimer information for information pertaining to code examples.

The following example will get the list of users in the target registry, MyRegistry, that are associated with the specified identifier.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>

void printAssociationType(int type);
void printListResults(EimList * list);
void printListData(char * fieldName,
                   void * entry,
                   int offset);


int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    EimIdentifierInfo x;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Set up identifier information */
    x.idtype = EIM_UNIQUE_NAME;
```

```
    x.id.uniqueName = "mjones";

    if (0 != (rc = eimGetTargetFromIdentifier(handle,
                                              &x,
                                              "MyRegistry",
                                              NULL,
                                              1000,
                                              list,
                                              err)))
    {
        printf("Get Target from identifier error = %d", rc);
        return -1;
    }

    printListResults(list);

    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));
        printAssociationType(entry->type);
        printListData("target group registry",
                      entry,
                      offsetof(EimTargetIdentity, targetGroupRegistry));

        /* advance to next entry */
        entry = (EimTargetIdentity *)((char *)entry + entry->nextEntry);

    }
    printf("\n");


}

void printListData(char * fieldName,
                   void * entry,
                   int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
```

```
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}

void printAssociationType(int type)
{
    switch(type)
    {
        case EIM_TARGET:
            printf("    Target Association.\n");
            break;
        case EIM_DEFAULT_DOMAIN_POLICY:
            printf("    Default Domain Policy Association.\n");
            break;
        default:
            printf("ERROR - unknown association type.\n");
            break;
    }
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

eimGetTargetFromSource() —Get EIM Target Identities from the Source

Syntax

```
#include <eim.h>

int eimGetTargetFromSource(EimHandle      * eim,
                           char           * sourceRegistryName,
                           char           * sourceRegistryUserName,
                           char           * targetRegistryName,
                           char           * additionalInformation,
                           unsigned int     lengthOfListData,
                           EimList        * listData,
                           EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimGetTargetFromSource()** function gets the target identity(ies) 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 return the user for this target registry.

» See EIM Mapping Lookup Algorithm for the steps involved in a mapping lookup operation. «

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator
- EIM Mapping Lookup
- EIM authority to an individual registry

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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

sourceRegistryName (Input)

The source registry for this lookup operation.

sourceRegistryUserName (Input)

The source user name for this lookup operation.

targetRegistryName (Input)

The target registry for this lookup operation. A NULL parameter indicates that the `localRegistry` set by the eimSetConfiguration() API or the eimSetConfigurationExt() API should be used.

additionalInfo (Input)

Additional information that will be used as selection criteria for this operation. This may be NULL. This filter data may contain the wild card char(*).

lengthOfListData (Input)

The number of bytes provided by the caller 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 API will return as much data as space has been provided. The data returned is a linked list of EimTargetIdentity structures. firstEntry is used to get to the first EimTargetIdentity structure in the linked list. Each EimTargetIdentity entry contains a user name returned by this lookup operation.

EimList structure:

```
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;
```

EimTargetIdentity structure:

```
typedef struct EimTargetIdentity
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure         */
    EimListData userName;           /* User name                       */
    enum EimAssociationType type;   /* Association type                */

    EimListData sourceGroupRegistry;/* Source group registry name      */
    EimListData targetGroupRegistry;/* Target group registry name      */
    EimSubList  credentialInfo;     /* EimCredentialInfo sublist       */

} EimTargetIdentity;
```

» The *sourceGroupRegistry* will be returned if the target identity was found using a source association to a group registry. The *targetGroupRegistry* will be returned if the target identity was found using a target association to a group registry.

This API will always return 0 for the number of entries in the credentialInfo sublist. If you need access to the credential information, use the Get EIM Target Identities and Credentials from the Source (eimGetTargetCredsFromSource) API.

EimSubList structure:

```
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;
                                       that is, the structure containing
                                       this structure.                */
} EimSubList;
```

«

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                 */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure; that is, the
                                       structure containing this
                                       structure.                     */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry not found or insufficient access to EIM data.

EIMERR_NOREG (28) EIM Registry not found or insufficient access to EIM data.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35) 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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23)	Unexpected LDAP error. %s
EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimGetTargetFromIdentifier() —Get EIM Target Identities from the Identifier" on page 128 —Get EIM Target Identities from the Identifier
- "eimGetTargetCredsFromSource() —Get EIM Target Identities and Credentials from the Source" on page 114 —Get EIM Target Identities and Credentials from the Source

Example

See Code disclaimer information for information pertaining to code examples.

The following example will get the target identity that is associated with the source inofirmation.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>


void printAssociationType(int type);
void printListResults(EimList * list);
void printListData(char * fieldName,
                   void * entry,
                   int offset);

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get target identity                      */
    if (0 != (rc = eimGetTargetFromSource(handle,
                                          "kerberosRegistry",
                                          "mjjones",
                                          "MyRegistry",
```

```
                                        NULL,
                                        1000,
                                        list,
                                        err)))
    {
        printf("Get Target from source error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    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));
        printAssociationType(entry->type);
        printListData("source group registry",
                      entry,
                      offsetof(EimTargetIdentity, sourceGroupRegistry));
        printListData("target group registry",
                      entry,
                      offsetof(EimTargetIdentity, targetGroupRegistry));

        /* advance to next entry */
        entry = (EimTargetIdentity *)((char *)entry + entry->nextEntry);

    }
    printf("\n");


}

void printListData(char * fieldName,
                   void * entry,
                   int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);
```

```
    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}

void printAssociationType(int type)
{
    switch(type)
    {
        case EIM_TARGET:
            printf("    Target Association.\n");
            break;
        case EIM_CERT_FILTER_POLICY:
            printf("    Certificate Filter Policy Association.\n");
            break;
        case EIM_DEFAULT_REG_POLICY:
            printf("    Default Registry Policy Association.\n");
            break;
        case EIM_DEFAULT_DOMAIN_POLICY:
            printf("    Default Domain Policy Association.\n");
            break;
        default:
            printf("ERROR - unknown association type.\n");
            break;
    }
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimGetVersion()—Get EIM Version

Syntax

```
#include <eim.h>

int eimGetVersion(EimHostInfo             * hostInfo,
                  enum EimVersion         * version,
                  EimRC                   * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimGetVersion()** function returns the EIM version supported by the local EIM APIs for the specified EIM host.

Authorities and Locks

None

Parameters

hostInfo (Input)

The structure that contains the EIM host information for which to return the EIM version supported by the local EIM APIs.

For EIM_HANDLE (0) host type, the *hostInfo* field must contain an EIM handle returned by a previous call to eimCreateHandle() and eimConnect().

For EIM_LDAP_URL (1) host type, the *hostInfo* field must contain a uniform resource locator (URL) that contains the EIM host information. A NULL value for the *ldapURL* field indicates that the ldap URL information set by the eimSetConfiguration() API should be used. This URL has the following format:

```
ldap://host:port/dn
      or
ldaps://host:port
```

where:

- *host:port* is the name of the host on which the EIM domain controller is running with an optional port number.
- *dn* is the distinguished name of the domain to work with (optional).
- *ldaps* indicates that this host/port combination uses SSL and TLS.

Examples:

- ldap://systemx:389/ibm-eimDomainName=myEimDomain,o=myCompany,c=us
- ldaps://systemy:636/ibm-eimDomainName=thisEimDomain

The structure layouts follow:

```
enum EimHostInfoType {
    EIM_HANDLE,
    EIM_LDAP_URL
};

typedef struct EimHostInfo
{
    enum EimHostInfoType  hostType;
    union {
        EimHandle       * eim;
        char            * ldapURL;
    } hostInfo;
} EimHostInfo;
```

version (Output)

The EIM version supported by the local EIM APIs for the specified host. Possible values are:

EIM_VERSION_0 (0)	EIM is not supported on the specified host.
EIM_VERSION_1 (1)	EIM version 1 is supported by the local EIM APIs for the specified host. This host supports EIM functionality provided with the first version of the EIM APIs .
EIM_VERSION_2 (2)	EIM version 2 is supported by the local EIM APIs for the specified host. This host supports EIM functionality provided with the second version of the EIM APIs, which includes policy association support.
EIM_VERSION_3 (3)	EIM version 3 is supported by the local EIM APIs for the specified host. This host supports EIM functionality provided with the third version of the EIM APIs, which includes credentials and group registries.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26)	Unable to allocate internal system object.

EINVAL
Input parameter was not valid.

EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
EIMERR_PARM_REQ (34)	Missing required parameter. Please check API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_URL_NOHOST (47)	URL does not have a host.
EIMERR_URL_NOTLDAP (49)	URL does not begin with ldap.
EIMERR_TYPE_INVAL (69)	The specified type is not valid.

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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23)	Unexpected LDAP error. %s
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.
EIMERR_LDAP_SCHEMA_NOT_FOUND (71)	Unable to find LDAP schema.

Example

See Code disclaimer information for information pertaining to code examples.

The following example gets the version supported by the local EIM APIs for the EIM host.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHostInfo   hostInfo;

    enum EimVersion version;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    hostInfo.hostInfo.eim = (EimHandle *)argv[1];
    hostInfo.hostType = EIM_HANDLE;

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get the version                          */
    if (0 != (rc = eimGetVersion(&hostInfo,
                                 &version,
                                 err)))
    {
        printf("Get Version error = %d", rc);
        return -1;
    }
    /* Print the version.                       */
    printf("Version = %d", version);

    return 0;
}
```

API introduced: V5R3

Top | Security APIs | APIs by category

---

eimListAccess()—List EIM Access

Syntax

```
#include <eim.h>

int eimListAccess(EimHandle          * eim,
                  enum EimAccessType   accessType,
                  char               * registryName,
                  unsigned int         lengthOfListData,
                  EimList            * listData,
      EimRC              * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimListAccess()** function lists the users that have the specified EIM access type.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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.

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

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.
EIM_ACCESS_CREDENTIAL_DATA (5)	Authority to retrieve credential data.

registryName (Input)

The name of the EIM registry for which access is to be listed. This parameter is only used if EimAccessType is EIM_ACCESS_REGISTRY.

lengthOfListData (Input)

The number of bytes provided by the caller 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 API will return as much data as space has been provided. The data returned is a linked list of EimAccess structures. firstEntry is used to get to the first EimAccess structure in the linked list.

EimList structure:

```
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;
```

EimAccess structure:

```
    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;
```

EimListData structure:

```
    typedef struct EimListData
    {
        unsigned int length;            /* Length of data                  */
        unsigned int disp;              /* Displacement to data.  This byte
                                        offset is relative to the start of
                                        the parent structure; that is, the
                                        structure containing this
                                        structure.                      */
    } EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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_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 API documentation.
EIMERR_PTR_INVAL (35)	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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23)	Unexpected LDAP error. %s
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimAddAccess()—Add EIM Access" on page 11 —Add EIM Access
- "eimRemoveAccess()—Remove EIM Access" on page 217 —Remove EIM Access
- "eimListUserAccess()—List EIM User Access" on page 204 —List EIM User Access
- "eimQueryAccess()—Query EIM Access" on page 210 —Query EIM Access

Example

See Code disclaimer information for information pertaining to code examples.

The following example lists all users with the specified access.

```
#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[100];
    EimRC       * err;
    EimHandle   * handle;

    char          listData[5000];
    EimList     * list = (EimList * ) listData;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* List all users with this access          */
    if (0 != (rc = eimListAccess(handle,
                                 EIM_ACCESS_ADMIN,
                                 NULL,
                                 5000,
                                 list,
                                 err)))
    {
        printf("List access error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    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");

}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimListAssociations()— List EIM Associations

Syntax

```
#include <eim.h>

int eimListAssociations(EimHandle              * eim,
                        enum EimAssociationType  associationType,
                        EimIdentifierInfo      * idName,
                        unsigned int             lengthOfListData,
                        EimList                * listData,
      EimRC                    * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimListAssociations()** function returns a list of associations for a given EIM identifier. This can be used to find all of the associated identities for an individual in the enterprise.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator

- EIM Mapping Lookup
- EIM authority to an individual registry

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

Parameters

eim (Input)
The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

associationType (Input)
The type of association to be listed. Valid values are:

EIM_ALL_ASSOC (0)	List all target, source, and administrative 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 whose associations are to be listed. 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` indicates which identifier name is provided. Use of the `uniqueName` provides the best performance. Specifying an `idtype` of EIM_NAME does not guarantee that a unique EIM identifier will be found. Therefore, use of EIM_NAME may result in an error.

lengthOfListData (Input)
The number of bytes provided by the caller 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 API will return as much data as space has been provided. The data returned is a linked list of EimAssociation structures. firstEntry is used to get to the first EimAssociation structure in the linked list.

EimList structure:

```
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;
```

EimAssociation structure:

```
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;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                */
    unsigned int disp;              /* Displacement to data.  This byte
                                    offset is relative to the start of
                                    the parent structure; that is, the
                                    structure containing this
                                    structure.                        */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Identifier name is not valid or insufficient access to EIM data.

EIMERR_IDNAME_AMBIGUOUS (20) More than 1 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) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EINVAL
Input parameter was not valid.

EIMERR_ASSOC_TYPE_INVAL (4) Association type is not valid.
EIMERR_EIMLIST_SIZE (16) Length of EimList is not valid. EimList must be at least 20 bytes in length.
EIMERR_HANDLE_INVAL (17) EimHandle is not valid.
EIMERR_IDNAME_TYPE_INVAL (52) The EimIdType value is not valid.
EIMERR_PARM_REQ (34) Missing required parameter. Please check API documentation.
EIMERR_PTR_INVAL (35) 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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s
EIMERR_UNEXP_OBJ_ VIOLATION (56) Unexpected object violation.
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimGetAssociatedIdentifiers() —Get Associated EIM identifiers" on page 91 —Get Associated EIM Identifiers
- "eimAddAssociation()—Add EIM Association" on page 15—Add an EIM Association
- "eimRemoveAssociation()— Remove EIM Association" on page 220—Remove an EIM Associations

Example

The following example will list the associations for an identifier.

See Code disclaimer information for information pertaining to code examples.

```
#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[100];
    EimRC       * err;
    EimHandle   * handle;

    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    EimIdentifierInfo x;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* 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,
                                       1000,
                                       list,
                                       err)))
    {
        printf("List Association error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);
```

```
    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");

}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimListDomains()—List EIM Domain Objects

Syntax

```
#include <eim.h>

int eimListDomains(char           * ldapURL,
                   EimConnectInfo   connectInfo,
                   unsigned int     lengthOfListData,
                   EimList        * listData,
       EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimListDomains()** function can be used to list information for a single EIM domain or list information for all EIM domains that are reachable from this platform in the network.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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.

Parameters

ldapURL (Input)

A uniform resource locator (URL) that contains the EIM host information. This URL has the following format:

```
ldap://host:port/dn
      or
ldaps://host:port/dn
```

where:

- `host:port` is the name of the host on which the EIM domain controller is running with an optional port number.
- `dn` is the distinguished name of the domain to list. If dn is not set then all domains that are reachable from this platform are returned.
- `ldaps` indicates that this host/port combination uses SSL and TLS.

Examples:

- ldap://systemx:389/ibm-eimDomainName=myEimDomain,o=myCompany,c=us
- ldaps://systemy:636/

connectInfo (Input)

Connect information. EIM uses ldap. 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 EIM_SIMPLE connect type, the `creds` field should contain the EimSimpleConnectInfo structure with a binddn and password. EimPasswordProtect is used to determine 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. EimSSLInfo must be provided.

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 provided by the caller for the list of domains. Minimum size required is 20 bytes. The API will return 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 be returned.

The EimList structure contains information about the returned data. The data returned is a linked list of EimDomain structures. firstEntry is used to get to the first EimDomain structure in the linked list.

EimList structure:

```
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 structure:

```
typedef struct EimDomain
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                     byte offset is relative to the
                                     start of this structure           */
    EimListData name;               /* Domain name                       */
    EimListData dn;                 /* Distinguished name for the domain
                                                                       */
    EimListData description;        /* Description                       */
    enum EimStatus  policyAssociations;  /* Policy associations
      attribute             */
} EimDomain;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                */
    unsigned int disp;              /* Displacement to data.  This byte
                                     offset is relative to the start of
                                     the parent structure; that is, the
                                     structure containing this
                                     structure.                         */
} EimListData;
```

eimrc (Input/Output)
The structure in which to return error code information. If the return value is not 0, eimrc will be set with additional information. This parameter may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

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

EIMERR_NODOMAIN (24) EIM Domain not found or insufficient access to EIM data.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

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 API documentation.
EIMERR_PROTECT_INVAL (22)	The protect parameter in EimSimpleConnectInfo is not valid.
EIMERR_PTR_INVAL (35)	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 (required).
EIMERR_URL_NOHOST (47)	URL does not have a host.
EIMERR_URL_NOTLDAP (49)	URL does not begin with ldap.
EIMERR_INVALID_DN (66)	Distinguished Name (DN) is not valid.

ENOMEM
Unable to allocate required space.

EIMERR_NOMEM (27) No memory available. Unable to allocate required space.

ENOTSUP
Connection type is not supported.

Connection type is not supported.

EIMERR_CONN_NOTSUPP (12)

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimDeleteDomain()—Delete an EIM Domain Object" on page 72—Delete an EIM Domain Object
- "eimCreateDomain()—Create an EIM Domain Object" on page 65—Create an EIM Domain Object
- "eimChangeDomain()—Change an EIM Domain Object" on page 36—Change an EIM Domain Object

Example

See Code disclaimer information for information pertaining to code examples.

The following example lists the information for the specified EIM domain.

```
#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[100];
    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,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;


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

}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimListIdentifiers()— List EIM Identifiers

Syntax

```
#include <eim.h>

int eimListIdentifiers(EimHandle         * eim,
                       EimIdentifierInfo * idName,
                       unsigned int        lengthOfListData,
                       EimList           * listData,
  EimRC             * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimListIdentifiers()** function returns a list of identifiers in the EIM domain. *idName* can be used to filter the results returned.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator
- EIM Mapping Lookup
- EIM authority to an individual registry

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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

idName (Input)

A structure that contains the name for this identifier. This parameter may be NULL in which case no filtering would be done by idName. 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 will indicate which identifier name has been provided. There is no guarantee that name will find a unique identifier. Therefore, use of name may result in multiple identifiers being returned. The id values, uniqueName, entryUUID and name may contain the wild card (*).

lengthOfListData (Input)

The number of bytes provided by the caller 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 API will return as much data as space has been provided. The data returned is a linked list of EimIdentifier structures. firstEntry is used to get to the first EimIdentifier structure in the linked list.

EimList structure:

```
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;
```

EimIdentifier structure:

```
typedef struct EimIdentifier
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure          */
```

```
    EimListData uniquename;         /* Unique name                   */
    EimListData description;        /* Description                   */
    EimListData entryUUID;          /* UUID                          */
    EimSubList  names;              /* EimIdentifierName sublist     */
    EimSubList  additionalInfo;     /* EimAddlInfo sublist           */
    enum EimAssociationType type;   /* Association type - not valid  */

    EimListData groupRegistry;      /* Group registry - not valid    */

} EimIdentifier;
```

Identifiers may have 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.

EimIdentifierName structure:

```
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.

EimAddlInfo structure:

```
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;
```

EimSubList structure:

```
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;
                                       that is, the structure containing
                                       this structure.               */
} EimSubList;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure; that is, the
                                       structure containing this
                                       structure.                    */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Identifier name is not valid or insufficient access to EIM data.

EIMERR_NOIDENTIFIER (25) EIM Identifier not found or insufficient access to EIM data.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35) 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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimAddIdentifier()—Add EIM Identifier" on page 23—Add EIM Identifier
- "eimChangeIdentifier()— Change EIM Identifier" on page 41—Change EIM Identifier
- "eimRemoveIdentifier()— Remove EIM Identifier" on page 224—Remove EIM Identifier
- "eimGetAssociatedIdentifiers() —Get Associated EIM identifiers" on page 91 —Get Associated EIM Identifiers

Example

See Code disclaimer information for information pertaining to code examples.

The following example will list 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[100];
    EimRC       * err;
    EimHandle   * handle;

    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get all identifiers                      */
    if (0 != (rc = eimListIdentifiers(handle,
                      NULL,
                      1000,
                      list,
```

```
                    err)))
    {
        printf("List identifiers error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    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");

}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimListPolicyFilters()—List EIM Policy Filters

Syntax

```
#include <eim.h>

int eimListPolicyFilters(EimHandle               * eim,
                         enum EimPolicyFilterType  filterType,
                         char                    * registryName,
                         unsigned int              lengthOfListData,
                         EimList                 * listData,
                         EimRC                   * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimListPolicyFilters()** function lists policy filters for the domain.

» EIM version 2 must be supported by the local EIM APIs to use this API (see "eimGetVersion()—Get EIM Version" on page 140—Get EIM Version). «

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator
- EIM Mapping Lookup
- EIM authority to an individual registry

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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

filterType (Input)

The type of policy filters to be listed. Valid values are:

EIM_ALL_FILTERS (0)	List all policy filters.
EIM_CERTIFICATE_FILTER (1)	List certificate policy filters.

registryName (Input)

The name of the registry for which to list policy filters. If a NULL value is specified, then policy filters for the entire domain will be listed.

lengthOfListData (Input)

The number of bytes provided by the caller 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 API will return as much data as space has been provided. The data returned is a linked list of EimPolicyFilter structures. firstEntry is used to get to the first EimPolicyFilter structure in the linked list.

EimList structure:

```
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;
```

EimPolicyFilter structure:

```
typedef struct EimPolicyFilter
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure          */
    enum EimPolicyFilterType type;  /* Type of policy filter.           */
    EimListData sourceRegistry;     /* Source registry name the policy
                                       filter is defined for.           */
    EimListData filterValue;        /* Policy filter value.             */
} EimPolicyFilter;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                   */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure; that is, the
                                       structure containing this
                                       structure.                       */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry name is not valid 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) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13)	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 API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_SPACE (41)	Unexpected error accessing parameter.
EIMERR_POLICY_FILTER_TYPE_INVAL (60)	Policy filter type is not valid.
EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified function is not supported by the EIM version.

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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23)	Unexpected LDAP error. %s
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.
EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.

Related Information

- "eimAddPolicyFilter()—Add EIM Policy Filter" on page 31 —Add EIM Policy Filter
- "eimRemovePolicyFilter()—Remove EIM Policy Filter" on page 232 —Remove EIM Policy Filter

Example

See Code disclaimer information for information pertaining to code examples.

The following example lists certificate policy filters for a registry.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>

void printPolicyFilterType(int type);
void printListResults(EimList * list);
void printListData(char * fieldName,
                   void * entry,
                   int offset);


int main (int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get source registry policies             */
    if (0 != (rc = eimListPolicyFilters(handle,
                                        EIM_CERTIFICATE_FILTER,
                                        "MySourceRegistry",
                                        1000,
                                        list,
                                        err)))
    {
        printf("List EIM Policy Filters error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    return 0;

}

void printListResults(EimList * list)
{
    int i;
    EimPolicyFilter * entry;

    printf("___________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimPolicyFilter *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
        printf("\n");
        printf("===============\n");
```

```
        printf("Entry %d.\n", i);

        /* Print out results */
        printPolicyFilterType(entry->type);
        printListData("Source Registry",
                      entry,
                      offsetof(EimPolicyFilter, sourceRegistry));
        printListData("Filter Value",
                      entry,
                      offsetof(EimPolicyFilter, filterValue));

        /* advance to next entry */
        entry = (EimPolicyFilter *)((char *)entry + entry->nextEntry);

    }
    printf("\n");


}

void printPolicyFilterType(int type)
{
    switch(type)
    {
        case EIM_CERTIFICATE_FILTER:
            printf("    Certificate Filter Policy.\n");
            break;
        default:
            printf("ERROR - unknown policy filter 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");

}
```

API introduced: V5R3

Top | Security APIs | APIs by category

eimListRegistries()—List EIM Registries

Syntax

```
#include <eim.h>

int eimListRegistries(EimHandle           * eim,
                      char                * registryName,
                      char                * registryType,
                      enum EimRegistryKind  registryKind,
                      unsigned int          lengthOfListData,
                      EimList             * listData,
                      EimRC               * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimListRegistries()** function lists the user registries participating in the EIM domain. The following parameters can be used to filter the results returned: registryType, registryName and registryKind.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator
- EIM Mapping Lookup
- EIM authority to an individual registry

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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

registryName (Input)

The name of the EIM registry to list. The name may contain the wild card char (*). This is used as a filter to determine which registries to return. This parameter may be NULL in which case no filtering would be done by name.

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. See eim.h for a list of defined types. This parameter may be NULL in which case no filtering would be done by type.

registryKind (Input)

The kind of registry to list. Valid values are:

EIM_ALL_REGISTRIES (0)	System, application, and group registries will be returned.
EIM_SYSTEM_REGISTRY (1)	Return system registries.
EIM_APPLICATION_REGISTRY (2)	Return application registries.
EIM_GROUP_REGISTRY (3)	Return group registries.

lengthOfListData (Input)

The number of bytes provided by the caller for the *listData* parameter. The minimum size required is 20 bytes.

listData (Output)

A pointer to the data to be returned.

The EimList structure contains information about the returned data. The API will return as much data as space has been provided. The data returned is a linked list of EimRegistry structures. firstEntry is used to get to the first EimRegistry structure in the linked list.

EimList structure:

```
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;
```

EimRegistry structure:

```
typedef struct EimRegistry
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure          */
    enum EimRegistryKind  kind;     /* Kind of registry                 */
    EimListData name;               /* Registry name                    */
    EimListData type;               /* Registry type                    */
    EimListData description;        /* Description                      */
    EimListData entryUUID;          /* Entry UUID                       */
    EimListData URI;                /* URI                              */
    EimListData systemRegistryName; /* System registry name             */
    EimSubList  registryAlias;      /* EimRegistryAlias sublist         */
    enum EimStatus  mappingLookup;  /* Mapping lookup attribute         */
    enum EimStatus  policyAssociations; /* Policy associations
         attribute                 */

    EimSubList  registryMembers;    /* EimRegistryName sublist          */
    EimSubList  registryGroups;     /* EimRegistryName sublist          */

} EimRegistry;
```

Registries may have a number of aliases defined. In the EimRegistry structure, the registryAlias EimSubList gives addressability to the first EimRegistryAlias structure.

EimRegistryAlias structure:

```
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;
```

Group registries may have a number of members defined. In the EimRegistry structure, the registryMembers EimSubList gives addressability to the first EimRegistryName structure. Registry members will only exist for registries that have a type of group registry.

Registries may have a number of group registries of which they are a member. In the EimRegistry structure, the registryGroups EimSubList gives addressability to the first EimRegistryName structure. Registry groups will only exist for registries that are not group registries.

EimRegistryName structure:

```
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;
```

EimSubList structure:

```
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;
                                     that is, the structure containing
                                     this structure.                  */
} EimSubList;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                 */
    unsigned int disp;              /* Displacement to data.  This byte
                                     offset is relative to the start of
                                     the parent structure; that is, the
                                     structure containing this
                                     structure.                       */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26)	Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13)	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 API documentation.
EIMERR_PTR_INVAL (35)	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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23)	Unexpected LDAP error. %s
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimAddSystemRegistry()—Add a System Registry to the EIM domain" on page 3 —Add a System Registry to the EIM Domain

- “eimAddApplicationRegistry()—Add an Application Registry to the EIM Domain” on page 7 —Add an Application Registry to the EIM Domain
- “eimAddGroupRegistry()—Add a Group Registry to the EIM domain” on page 19 —Add a Group Registry to the EIM Domain
- “eimRemoveRegistry()—Remove a Registry from the EIM Domain” on page 214 —Remove a Registry from the EIM Domain
- “eimChangeRegistry()—Change EIM Registry” on page 44 —Change EIM Registry

Example

See Code disclaimer information for information pertaining to code examples.

The following example lists all registries found.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>


void printRegistryKind(int kind);
void printListResults(EimList * list);
void printListData(char * fieldName,
                   void * entry,
                   int offset);
void printAliasSubList(void * entry,
                       int offset);
void printNameSubList(void * entry,
                      int offset);


int main (int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get all registries                       */
    if (0 != (rc = eimListRegistries(handle,
                                     NULL,
                                     NULL,
                                     EIM_ALL_REGISTRIES,
                                     1000,
                                     list,
                                     err)))
    {
        printf("List registries error = %d", rc);
        return -1;
    }
```

```
    /* Print the results                        */
    printListResults(list);

    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));
        printf("List of member registries:\n");
        printNameSubList(entry,
                         offsetof(EimRegistry, registryMembers));
        printf("List of group registries:\n");
        printNameSubList(entry,
                         offsetof(EimRegistry, registryGroups));

        /* 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;
        case EIM_GROUP_REGISTRY:
            printf("Group Registry.\n");
            break;
        default:
            printf("ERROR - unknown registry kind.\n");
            break;
    }
}

void printListData(char * fieldName,
                   void * entry,
                   int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}


void printAliasSubList(void * entry,
                       int offset)
{
    int i;
    EimSubList * subList;
    EimRegistryAlias * subentry;

    /* Address the EimSubList object */
    subList = (EimSubList *)((char *)entry + offset);

    if (subList->listNum > 0)
    {

        subentry = (EimRegistryAlias *)((char *)entry +
                                        subList->disp);
        for (i = 0; i < subList->listNum; i++)
        {

            /* Print out results */
            printListData("Registry alias type",
                          subentry,
                          offsetof(EimRegistryAlias, type));
            printListData("Registry alias value",
                          subentry,
                          offsetof(EimRegistryAlias, value));
```

```
            /* advance to next entry */
            subentry = (EimRegistryAlias *)((char *)subentry +
                                            subentry->nextEntry);
        }
    }

}


void printNameSubList(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("Registry name",
                          subentry,
                          offsetof(EimRegistryName, name));

            /* advance to next entry */
            subentry = (EimRegistryName *)((char *)subentry +
                                           subentry->nextEntry);
        }
    }

}
```

API introduced: V5R2

Top | Security APIs | APIs by category

eimListRegistryAliases()—List EIM Registry Aliases

Syntax

```
#include <eim.h>

int eimListRegistryAliases(EimHandle      * eim,
                           char           * registryName,
                           unsigned int     lengthOfListData,
                           EimList        * listData,
         EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimListRegistriesAliases()** function returns a list of all the aliases defined for a particular registry.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator
- EIM Mapping Lookup
- EIM authority to an individual registry

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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

registryName (Input)

The name of the registry for which to list aliases.

lengthOfListData (Input)

The number of bytes provided by the caller for the *listData* parameter. The minimum size required is 20 bytes.

listData (Output)

A pointer to the data to be returned.

The EimList structure contains information about the returned data. The API will return as much data as space has been provided. The data returned is a linked list of EimRegistryAlias structures. firstEntry is used to get to the first EimRegistryAlias structure in the linked list.

EimList structure:

```
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;
```

EimRegistryAlias structure:

```
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;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;           /* Length of data                */
    unsigned int disp;             /* Displacement to data.  This byte
                                    offset is relative to the start of
                                    the parent structure; that is, the
                                    structure containing this
                                    structure.                       */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry not found or insufficient access to EIM data.

EIMERR_NOREG (28) EIM Registry not found or insufficient access to EIM data.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.

EIMERR_PTR_INVAL (35) 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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimChangeRegistryAlias()—Change EIM Registry Alias" on page 49 —Change EIM Registry Alias
- "eimGetRegistryNameFromAlias() —Get EIM Registry Name from an Alias" on page 109 —Get EIM Registry Name from an Alias

Example

See Code disclaimer information for information pertaining to code examples.

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[100];
    EimRC       * err;
    EimHandle   * handle;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;
```

```
    /* Get all aliases for the registry         */
    if (0 != (rc = eimListRegistryAliases(handle,
                                          "MyRegistry",
                                          1000,
                                          list,
                                          err)))
    {
        printf("List registry aliases error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    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");

}
```

API introduced: V5R2

---

eimListRegistryAssociations()—List EIM Registry Associations

Syntax

```
#include <eim.h>

int eimListRegistryAssociations(EimHandle               * eim,
                                enum EimAssociationType   associationType,
                                char                    * registryName,
                                char                    * registryUserName,
                                unsigned int              lengthOfListData,
                                EimList                 * listData,
                                EimRC                   * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimListRegistryAssociations()** function lists association information for the registry or domain.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator
- EIM Mapping Lookup
- EIM authority to an individual registry

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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

associationType (Input)

The type of associations to be listed.

» EIM version 2 must be supported by the local EIM APIs to specify a policy association type (see "eimGetVersion()—Get EIM Version" on page 140—Get EIM Version). « 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 source and target associations.
EIM_ADMIN (4)	List administrative associations.
EIM_ALL_POLICY_ASSOC (5)	List all policy associations.
EIM_CERT_FILTER_POLICY (6)	List certificate filter policy associations.
EIM_DEFAULT_REG_POLICY (7)	List default registry policy associations.
EIM_DEFAULT_DOMAIN_POLICY (8)	List default domain policy associations.

registryName (Input)

The name of the registry for which to list association information. If a NULL value is specified, then association information for the entire domain will be listed. The *registryUserName* parameter must also be NULL if this parameter is NULL.

If the *registryUserName* parameter is NULL, then for policy associations, the association information will include policy associations where the registry is the source registry or the target registry. If the *registryUserName* parameter is not NULL, then the association information will include policy associations where the registry is the target registry.

registryUserName (Input)

The name of the registry user name for which to list association information.

lengthOfListData (Input)

The number of bytes provided by the caller for the *listData* parameter. 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 API will return as much data as space has been provided. The data returned is a linked list of EimRegistryAssociation structures. firstEntry is used to get to the first EimRegistryAssociation structure in the linked list.

EimList structure:

```
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;
```

EimRegistryAssociation structure:

```
typedef struct EimRegistryAssociation
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure          */
    enum EimAssociationType type;   /* Type of association.             */
    EimListData registryName;       /* Registry name the association
                                       is defined to.                   */
    EimListData registryUserName;   /* Registry user name the
                                       association is defined to.       */
```

```
    EimListData identifier;         /* Unique name for eim identifier */
    EimListData sourceRegistry;     /* Source registry name the
                                     association is defined for.      */
    EimListData filterValue;        /* Filter value                     */
    enum EimStatus domainPolicyAssocStatus;
                                    /* Policy association status for
                                     the domain:
                                     0 = not enabled
                                     1 = enabled                      */
    enum EimStatus sourceMappingLookupStatus;
                                    /* Mapping lookup status for the
                                     source registry:
                                     0 = not enabled
                                     1 = enabled                      */
    enum EimStatus targetMappingLookupStatus;
                                    /* Mapping lookup status for the
                                     target registry:
                                     0 = not enabled
                                     1 = enabled                      */
    enum EimStatus targetPolicyAssocStatus;
                                    /* Policy association status for
                                     the target registry:
                                     0 = not enabled
                                     1 = enabled                      */
} EimRegistryAssociation;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                */
    unsigned int disp;              /* Displacement to data.  This byte
                                     offset is relative to the start of
                                     the parent structure; that is, the
                                     structure containing this
                                     structure.                       */
} EimListData;
```

If the *type* field is EIM_TARGET (1), then the following fields will be returned:

- registryName
- registryUserName
- identifier
- targetMappingLookupStatus

If the *type* field is EIM_SOURCE (2), then the following fields will be returned:

- registryName
- registryUserName
- identifier
- sourceMappingLookupStatus

If the *type* field is EIM_ADMIN (4), then the following fields will be returned:

- registryName
- registryUserName
- identifier

If the *type* field is EIM_CERT_FILTER_POLICY (6), then the following fields will be returned:

- registryName
- registryUserName
- sourceRegistry
- filterValue

- domainPolicyAssocStatus
- sourceMappingLookupStatus
- targetMappingLookupStatus
- targetPolicyAssocStatus

If the *type* field is EIM_DEFAULT_REG_POLICY (7), then the following fields will be returned:

- registryName
- registryUserName
- sourceRegistry
- domainPolicyAssocStatus
- sourceMappingLookupStatus
- targetMappingLookupStatus
- targetPolicyAssocStatus

If the *type* field is EIM_DEFAULT_DOMAIN_POLICY (8), then the following fields will be returned:

- registryName
- registryUserName
- domainPolicyAssocStatus
- targetMappingLookupStatus
- targetPolicyAssocStatus

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry name is not valid 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) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13)	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 API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_SPACE (41)	Unexpected error accessing parameter.
EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified function is not supported by the EIM version.

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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23)	Unexpected LDAP error. %s
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.
EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.

Related Information

- "eimAddPolicyAssociation()—Add EIM Policy Association" on page 26 —Add EIM Policy Association
- "eimAddAssociation()—Add EIM Association" on page 15 —Add EIM Association
- "eimGetAssociatedIdentifiers() —Get Associated EIM identifiers" on page 91 —Get Associated EIM identifiers

Example

See Code disclaimer information for information pertaining to code examples.

The following example lists the default registry policies for a registry.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>

void printAssociationType(int type);
void printStatus(int status);
void printListResults(EimList * list);
void printListData(char * fieldName,
                   void * entry,
                   int offset);


int main (int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get source registry policies             */
    if (0 != (rc = eimListRegistryAssociations(handle,
                                               EIM_DEFAULT_REG_POLICY,
                                               "MySourceRegistry",
                                               NULL,
                                               1000,
                                               list,
                                               err)))
    {
        printf("List EIM Registry Associations error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    return 0;

}

void printListResults(EimList * list)
{
    int i;
    EimRegistryAssociation * entry;

    printf("__________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("   entriesReturned  = %d\n", list->entriesReturned);
    printf("   entriesAvailable = %d\n", list->entriesAvailable);
    printf("\n");

    entry = (EimRegistryAssociation *)((char *)list + list->firstEntry);
    for (i = 0; i < list->entriesReturned; i++)
    {
```

```
        printf("\n");
        printf("===============\n");
        printf("Entry %d.\n", i);

        /* Print out results */
        printAssociationType(entry->type);
        printListData("Registry Name",
                      entry,
                      offsetof(EimRegistryAssociation, registryName));
        printListData("Registry User Name",
                      entry,
                      offsetof(EimRegistryAssociation, registryUserName));
        printListData("EIM identifier",
                      entry,
                      offsetof(EimRegistryAssociation, identifier));
        printListData("Source Registry",
                      entry,
                      offsetof(EimRegistryAssociation, sourceRegistry));
        printListData("Filter Value",
                      entry,
                      offsetof(EimRegistryAssociation, filterValue));
        printStatus("Domain policy association status",entry->domainPolicyAssocStatus);
        printStatus("Source registry mapping lookup status",entry->sourceMappingLookupStatus);
        printStatus("Target registry mapping lookup status",entry->targetMappingLookupStatus);
        printStatus("Target registry policy association status",entry->targetPolicyAssocStatus);


        /* advance to next entry */
        entry = (EimRegistryAssociation *)((char *)entry + entry->nextEntry);

    }
    printf("\n");


}

void printAssociationType(int type)
{
    switch(type)
    {
        case EIM_TARGET:
            printf("    Target Association.\n");
            break;
        case EIM_SOURCE:
            printf("    Source Association.\n");
            break;
        case EIM_ADMIN:
            printf("    Administrative Association.\n");
            break;
        case EIM_CERT_FILTER_POLICY:
            printf("    Certificate Filter Policy Association.\n");
            break;
        case EIM_DEFAULT_REG_POLICY:
            printf("    Default Registry Policy Association.\n");
            break;
        case EIM_DEFAULT_DOMAIN_POLICY:
            printf("    Default Domain Policy Association.\n");
            break;
        default:
            printf("ERROR - unknown association type.\n");
            break;
    }
}

void printStatus(char * fieldName,
                 int    status)
{
```

```
    printf("     %s = ",fieldName);
    switch(status)
    {
        case EIM_STATUS_NOT_ENABLED:
            printf("    Not enabled.\n");
            break;
        case EIM_STATUS_ENABLED:
            printf("    Enabled.\n");
            break;
        default:
            printf("ERROR - unknown status value.\n");
            break;
    }
}

void printListData(char * fieldName,
                   void * entry,
                   int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;

    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}
```

API introduced: V5R3

Top | Security APIs | APIs by category

---

eimListRegistryUsers()— List EIM Registry Users

Syntax

```
#include <eim.h>

int eimListRegistryUsers(EimHandle      * eim,
                         char           * registryName,
                         char           * registryUserName,
                         unsigned int     lengthOfListData,
                         EimList        * listData,
                         EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimListRegistryUsers()** function lists the users in a particular registry that have target associations defined.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator
- EIM Mapping Lookup
- EIM authority to an individual registry

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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

registryName (Input)

The name of the registry that contains this user.

registryUserName (Input)

The name of the user in this registry to list. NULL will indicate all users.

lengthOfListData (Input)

The number of bytes provided by the caller 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 API will return as much data as space has been provided. The data returned is a linked list of EimRegistryUser structures. firstEntry is used to get to the first EimRegistryUser structure in the linked list.

EimList structure:

```
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;
```

EimRegistryUser structure:

```
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           */

    EimSubList  credentialInfo;     /* EimCredentialInfo sublist     */

} EimRegistryUser;
```

Registry users may have several additional attributes. In the EimRegistryUser structure, additionalInfo gives addressability to the first EimAddlInfo structure that contains a linked list of attributes.

EimAddlInfo structure:

```
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;
```

» This API will always return 0 for the numbers of entries in the credentialInfo sublist. If you need access to the credential information, use the List EIM Registry Users Credentials (eimListRegistryUsersCreds) API. «

EimSubList structure:

```
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;
                                       that is, the structure containing
                                       this structure.                  */
} EimSubList;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                 */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure; that is, the
                                       structure containing this
                                       structure.                       */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry not found or insufficient access to EIM data.

EIMERR_NOREG (28) EIM Registry not found or insufficient access to EIM data.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35) 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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23)	Unexpected LDAP error. %s
EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimChangeRegistryUser() —Change EIM Registry User" on page 52—Change EIM Registry User
- "eimListRegistryUsersCreds()— List EIM Registry Users Credentials" on page 197—List EIM Registry Users Credentials

Example

See Code disclaimer information for information pertaining to code examples.

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[100];
    EimRC       * err;
    EimHandle   * handle;

    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get registry user                        */
    if (0 != (rc = eimListRegistryUsers(handle,
                                        "MyRegistry",
                                        NULL,
                                        1000,
                                        list,
                                        err)))
    {
        printf("List registry users error = %d", rc);
        return -1;
    }

    /* Print the results                        */
```

```
    printListResults(list);

    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");

}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimListRegistryUsersCreds()— List EIM Registry Users Credentials

Syntax

```
#include <eim.h>

int eimListRegistryUsersCreds(EimHandle      * eim,
                              char           * registryName,
                              char           * registryUserName,
                              unsigned int     lengthOfListData,
                              EimList        * listData,
                              EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimListRegistryUsersCreds()** function lists the users in a particular registry that have target associations defined.

EIM version 3 must be supported by the local EIM APIs to use this API (see "eimGetVersion()—Get EIM Version" on page 140—Get EIM Version).

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to

EIM data. The access groups whose members have authority to the general registry user data (registry user name, description, and additional information) for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM Identifiers Administrator
- EIM Mapping Lookup
- EIM authority to an individual registry

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

The credential information for the registry user is considered security sensitive data. Access to this data is more strictly controlled. The access groups whose members have authority to the credential information for the registry user follow:

- EIM Administrator
- EIM Credential Data
- EIM authority to an individual registry

Note that the EIM Credential Data access group does not have access to the general registry user data. If a user is a member of the EIM Credential Data access group, then the user must also be a member of one of the access groups that has access to the general registry user data.

Parameters

eim (Input)
The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

registryName (Input)
The name of the registry that contains this user.

registryUserName (Input)
The name of the user in this registry to list. NULL will indicate all users.

lengthOfListData (Input)
The number of bytes provided by the caller 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 API will return as much data as space has been provided. The data returned is a linked list of EimRegistryUser structures. firstEntry is used to get to the first EimRegistryUser structure in the linked list.

EimList structure:

```
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;
```

EimRegistryUser structure:

```
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              */
    EimSubList  credentialInfo;     /* EimCredentialInfo sublist        */
} EimRegistryUser;
```

Registry users may have several additional attributes. In the EimRegistryUser structure, additionalInfo gives addressability to the first EimAddlInfo structure that contains a linked list of attributes.

EimAddlInfo structure:

```
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;
```

Registry users may have several types of credentials. In the EimRegistryUser structure, credentialInfo gives addressability to the first EimCredentialInfo structure that contains a linked list of credentials.

If there is credential information for the registry user, but the caller is not authorized to access the credential information, the EimCredentialInfo structure will be returned with the *type* and *status* fields filled in. The *data* field will not be returned (*length* and *disp* will be 0). If there is no credential information, the EimCredentialInfo structure will not be returned in the *credentialInfo* sublist.

EimCredentialInfo structure:

```
typedef struct EimCredentialInfo
{
    unsigned int nextEntry;         /* Displacement to next entry.  This
                                       byte offset is relative to the
                                       start of this structure.         */
    enum EimCredentialType type;    /* Credential type                  */
    enum EimStatus         status;  /* Credential status
                                       0 = not enabled
                                       1 = enabled                      */
    EimListData            data;    /* Credential data                  */
} EimCredentialInfo;
```

EimSubList structure:

```
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;
                                       that is, the structure containing
                                       this structure.                */
} EimSubList;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;           /* Length of data                */
    unsigned int disp;             /* Displacement to data.  This byte
                                    offset is relative to the start of
                                    the parent structure; that is, the
                                    structure containing this
                                    structure.                       */
} EimListData;
```

eimrc (Input/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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry not found or insufficient access to EIM data.

EIMERR_NOREG (28) EIM Registry not found or insufficient access to EIM data.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_SPACE (41)	Unexpected error accessing parameter.
EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified function is not supported by the EIM version.

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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23)	Unexpected LDAP error. %s
EIMERR_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimChangeRegistryUser() —Change EIM Registry User" on page 52—Change EIM Registry User
- "eimListRegistryUsers()— List EIM Registry Users" on page 191—List EIM Registry Users

Example

See Code disclaimer information for information pertaining to code examples.

The following example lists all users and credentials 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 printCredSubListData(char * fieldName,
                          void * entry,
                          int offset);
void printListData(char * fieldName,
                   void * entry,
                   int offset);


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

```
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    char          listData[1000];
    EimList     * list = (EimList * ) listData;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get registry user                        */
    if (0 != (rc = eimListRegistryUsersCreds(handle,
                                             "MyRegistry",
                                             NULL,
                                             1000,
                                             list,
                                             err)))
    {
        printf("List registry users credentials error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);

    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));
        printCredSubListData("Credential information",
                             entry,
```

```
                            offsetof(EimRegistryUser, credentialInfo));

        /* 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 printCredSubListData(char * fieldName,
                          void * entry,
                          int offset)
{
    int i;
    EimSubList * subList;
    EimCredentialInfo * subentry;

    /* Address the EimSubList object */
    subList = (EimSubList *)((char *)entry + offset);

    if (subList->listNum > 0)
    {
        subentry = (EimCredentialInfo *)((char *)entry + subList->disp);
        for (i = 0; i < subList->listNum; i++)
        {

            /* Print out results */
            printf("     Credential type = %d.\n",subentry->type);
            printf("     Credential status = %d.\n",subentry->status);
            /* Credential data is not printed.                */

            /* advance to next entry */
            subentry = (EimCredentialInfo *)((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");

}
```

API introduced: V5R4

Top | Security APIs | APIs by category

---

eimListUserAccess()—List EIM User Access

Syntax

```
#include <eim.h>

int eimListUserAccess(EimHandle      * eim,
                      EimAccessUser  * accessUser,
                      unsigned int     lengthOfListData,
                      EimList        * listData,
          EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimListUserAccess()** function lists the access groups of which this user is a member.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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.

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

accessUser (Input)

A structure that contains the user information for which to retrieve access.

EIM_ACCESS_LOCAL_USER	Indicates a local user name on the system that the API is run. The local user name will be converted to the appropriate access id for this system.
EIM_ACCESS_KERBEROS	Indicates a kerberos principal. The kerberos principal will be converted to the appropriate access id. For example, petejones@therealm will be 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;
```

lengthOfListData (Input)

The number of bytes provided by the caller 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 API will return as much data as space has been provided. The data returned is a linked list of EimUserAccess structures. firstEntry is used to get to the first EimUserAccess structure in the linked list.

EimList structure:

```
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;
```

EimUserAccess structure:

```
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        */

    enum EimAccessIndicator eimCredentialData;

} EimUserAccess;
```

The registries EimSubList gives addressability to a linked list of EimRegistryName structures.

EimRegistryName structure:

```
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;
```

EimSubList structure:

```
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;
                                     that is, the structure containing
                                     this structure.                  */
} EimSubList;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                 */
    unsigned int disp;              /* Displacement to data.  This byte
                                     offset is relative to the start of
                                     the parent structure; that is, the
                                     structure containing this
                                     structure.                       */
} EimListData;
```

eimrc (Input)
The structure in which to return error code information. If the return value is not 0, eimrc is set with additional information. This parameter may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35) 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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- “eimAddAccess()—Add EIM Access” on page 11 —Add EIM Access
- “eimRemoveAccess()—Remove EIM Access” on page 217 —Remove EIM Access
- “eimListAccess()—List EIM Access” on page 143 —List EIM User Accesses
- “eimQueryAccess()—Query EIM Access” on page 210 —Query EIM Access

Example

See Code disclaimer information for information pertaining to code examples.

The following example lists all registries found.

```
#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[100];
    EimRC       * err;
    EimHandle   * handle;

    EimAccessUser user;

    char          listData[5000];
    EimList     * list = (EimList * ) listData;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;


    /* 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,
                                     5000,
                                     list,
                                     err)))
    {
        printf("List user access error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printListResults(list);
```

```
    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");
    if (EIM_ACCESS_YES == entry->eimCredentialData)
        printf("     EIM Credential Data.\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");

}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimQueryAccess()—Query EIM Access

Syntax

```
#include <eim.h>

int eimQueryAccess(EimHandle          * eim,
                   EimAccessUser      * accessUser,
                   enum EimAccessType   accessType,
                   char               * registryName,
                   unsigned int       * accessIndicator,
                   EimRC              * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimQueryAccess()** function queries to see if the user has the specified access.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

eim (Input)
The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

accessUser (Input)
A structure that contains the user information for which to query access.

EIM_ACCESS_LOCAL_USER	Indicates a local user name on the system that the API is run. The local user name will be converted to the appropriate access id for this system.
EIM_ACCESS_KERBEROS	Indicates a kerberos principal. The kerberos principal will be converted to the appropriate access id. For example, petejones@therealm will be 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 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.
EIM_ACCESS_CREDENTIAL_DATA (5)	Authority to retrieve credential data.

registryName (Input)
The name of the EIM registry for which to check access. This parameter is only used if EimAccessType is EIM_ACCESS_REGISTRY.

accessIndicator (Output)
Indicator set to indicate if access found.

EIM_ACCESS_NO (0)	Access not found
EIM_ACCESS_YES (1)	Access found.

eimrc (Input/Output)
(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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35) 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 eimConnect() API and try the request again.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimAddAccess()—Add EIM Access" on page 11 —Add EIM Access
- "eimRemoveAccess()—Remove EIM Access" on page 217 —Remove EIM Access
- "eimListUserAccess()—List EIM User Access" on page 204 —List EIM User Access
- "eimListAccess()—List EIM Access" on page 143 —List EIM Access

Example

See Code disclaimer information for information pertaining to code examples.

The following example checks to see if the user has the requested access.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    EimAccessUser user;

    unsigned int indicator;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Set up access user info                  */
    user.userType = EIM_ACCESS_DN;
    user.user.dn="cn=pete,o=ibm,c=us";

    /* Query access for this user.              */
    if (0 != (rc = eimQueryAccess(handle,
                                  &user,
                                  EIM_ACCESS_ADMIN,
                                  NULL,
                                  &indicator,
                                  err)))
    {
        printf("Query access error = %d", rc);
        return -1;
    }
```

```
    /* Print the results                        */
    if (EIM_ACCESS_YES == indicator)
        printf("Access found\n");
    else
        printf("Access not found\n");

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimRemoveRegistry()—Remove a Registry from the EIM Domain

Syntax

```
#include <eim.h>

int eimRemoveRegistry(EimHandle      * eim,
                      char           * registryName,
          EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimRemoveRegistry()** function removes a currently participating registry from the EIM domain. It is recommended that all associations be removed for this registry before it is removed or it may result in orphaned associations. This includes admin, source, target, and policy associations. A system registry cannot be removed if there are any application registries that are a subset of this system registry.

When a registry is removed, an attempt is made to remove all associations for the registry. For policy associations, this would include any policy associations where this registry is the source registry or the target registry. If there are any policy filters defined for the registry, the policy filters will be removed along with any policy associations to the policy filters.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

registryName (Input)

The name of the registry to remove.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry not found or insufficient access to EIM data.

EIMERR_NOREG (28) EIM Registry not found or insufficient access to EIM data.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35) Pointer parameter is not valid.

ENOMEM
Unable to allocate required space.

EIMERR_NOMEM (27) No memory available. Unable to allocate required space.

ENOTCONN
LDAP connection has not been made.

EIMERR_NOT_CONN (31) Not connected to LDAP. Use eimConnect() 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 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) 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_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimAddSystemRegistry()—Add a System Registry to the EIM domain" on page 3 —Add a System Registry to the EIM Domain
- "eimAddApplicationRegistry()—Add an Application Registry to the EIM Domain" on page 7 —Add an Application Registry to the EIM Domain
- "eimAddGroupRegistry()—Add a Group Registry to the EIM domain" on page 19 —Add a Group Registry to the EIM Domain
- "eimChangeRegistry()—Change EIM Registry" on page 44 —Change EIM Registry
- "eimListRegistries()—List EIM Registries" on page 171 —List EIM Registries

Example

See Code disclaimer information for information pertaining to code examples.

The following example removes an EIM registry.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
```

```
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Remove the registry                      */
    if (0 != (rc = eimRemoveRegistry(handle,
                                     "MyRegistry",
                                     err)))
        printf("Remove registry error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimRemoveAccess()—Remove EIM Access

Syntax

```
#include <eim.h>

int eimRemoveAccess(EimHandle          * eim,
                    EimAccessUser      * accessUser,
                    enum EimAccessType   accessType,
                    char               * registryName,
      EimRC              * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimRemoveAccess()** function removes the user from the EIM access group identified by the access type.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

accessUser (Input)

A structure that contains the user information to remove access from.

EIM_ACCESS_LOCAL_USER	Indicates a local user name on the system that the API is run. The local user name will be converted to the appropriate access id for this system.
EIM_ACCESS_KERBEROS	Indicates a kerberos principal. The kerberos principal will be converted to the appropriate access id. For example, petejones@therealm will be 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 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.
» EIM_ACCESS_CREDENTIAL_DATA (5)	Authority to retrieve credential data. «

registryName (Input)
The name of the registry to remove access from. This parameter is only used if EimAccessType is EIM_ACCESS_REGISTRY. If EimAccessType is anything other than EIM_ACCESS_REGISTRY, this parameter must be NULL.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1)	Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26)	Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13)	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 API documentation.
EIMERR_PTR_INVAL (35)	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 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.

Related Information

- "eimAddAccess()—Add EIM Access" on page 11 —Add EIM Access
- "eimListAccess()—List EIM Access" on page 143 —List EIM Access
- "eimListUserAccess()—List EIM User Access" on page 204 —List EIM User Access
- "eimQueryAccess()—Query EIM Access" on page 210 —Query EIM Access

Example

See Code disclaimer information for information pertaining to code examples.

The following example removes the user from the access group.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    EimAccessUser user;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Set user information                     */
    user.userType = EIM_ACCESS_DN;
    user.user.dn="cn=pete,o=ibm,c=us";

    /* Remove access for this user.             */
    if (0 != (rc = eimRemoveAccess(handle,
                                &user,
                                EIM_ACCESS_ADMIN,
                                NULL,
                                err)))
    {
        printf("Remove access error = %d", rc);
        return -1;
    }

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimRemoveAssociation()— Remove EIM Association

Syntax

```
#include <eim.h>

int eimRemoveAssociation(EimHandle              * eim,
                         enum EimAssociationType  associationType,
                         EimIdentifierInfo      * idName,
                         char                   * registryName,
                         char                   * registryUserName,
                         EimRC                  * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimRemoveAssociation()** function removes an association for a local identity in a specified user registry with an EIM identifier.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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 authority to an individual registry

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

associationType (Input)

The type of association to be removed. Valid values are:

EIM_ALL_ASSOC (0)	Remove all target, source, and administrative 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 to remove this association from. 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` indicates which identifier name is provided. Use of the `uniqueName` provides the best performance. Specifying an `idtype` of EIM_NAME does not guarantee that a unique EIM identifier will be found. Therefore, use of EIM_NAME may result in an error.

registryName (Input)
The registry name.

registryUserName (Input)
The registry user name. The registry user name may be normalized according to the normalization method for defined registry.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry or identifier name is not valid or insufficient access to EIM data.

EIMERR_IDNAME_AMBIGUOUS (20) More than 1 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) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.

ENOMEM
Unable to allocate required space.

EIMERR_NOMEM (27)	No memory available. Unable to allocate required space.

ENOTCONN
LDAP connection has not been made.

EIMERR_NOT_CONN (31)	Not connected to LDAP. Use 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_UNEXP_OBJ_VIOLATION (56)	Unexpected object violation.
EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimGetAssociatedIdentifiers() —Get Associated EIM identifiers" on page 91 —Get Associated EIM Identifiers
- "eimAddAssociation()—Add EIM Association" on page 15—Remove an EIM Association
- "eimListAssociations()— List EIM Associations" on page 148—List EIM Associations

Example

See Code disclaimer information for information pertaining to code examples.

The following example will removes 2 associations.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
```

```
    EimHandle   * handle;

    EimIdentifierInfo x;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Set up identifier information.           */
    x.idtype = EIM_UNIQUE_NAME;
    x.id.uniqueName = "mjones";

    /* Remove association                       */
    if (0 != (rc = eimRemoveAssociation(handle,
                                    EIM_ADMIN,
                                    &x,
                                    "MyRegistry",
                                    "maryjones",
                                    err)))
    {
        printf("Remove Association error = %d", rc);
        return -1;
    }
    /* Remove association                       */
    if (0 != (rc = eimRemoveAssociation(handle,
                                    EIM_SOURCE,
                                    &x,
                                    "kerberosRegistry",
                                    "mjjones",
                                    err)))
    {
        printf("Remove Association error = %d", rc);
        return -1;
    }
    /* Remove association                       */
    if (0 != (rc = eimRemoveAssociation(handle,
                                    EIM_TARGET,
                                    &x,
                                    "MyRegistry",
                                    "maryjo",
                                    err)))
    {
        printf("Remove Association error = %d", rc);
        return -1;
    }

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimRemoveIdentifier()— Remove EIM Identifier

Syntax

```
#include <eim.h>

int eimRemoveIdentifier(EimHandle         * eim,
                        EimIdentifierInfo * idName,
      EimRC             * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimRemoveIdentifier()** function removes an EIM identifier and all of its associated mappings from the EIM domain.

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

idName (Input)

A structure that contains the name for this identifier. 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` will indicate which identifier name has been provided. Use of the `uniqueName` will provide the best performance. There is no guarantee that `name` will find a unique identifier. Therefore, use of `name` may result in an error.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Identifier not found or insufficient access to EIM data.

EIMERR_IDNAME_AMBIGUOUS (20) More than 1 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) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35) Pointer parameter is not valid.

ENOMEM
Unable to allocate required space.

EIMERR_NOMEM (27) No memory available. Unable to allocate required space.

ENOTCONN
LDAP connection has not been made.

EIMERR_NOT_CONN (31) Not connected to LDAP. Use 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_UNEXP_OBJ_VIOLATION (56) Unexpected object violation.
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimAddIdentifier()—Add EIM Identifier" on page 23—Add EIM Identifier
- "eimChangeIdentifier()— Change EIM Identifier" on page 41—Change EIM Identifier
- "eimListIdentifiers()— List EIM Identifiers" on page 160—List EIM Identifiers
- "eimGetAssociatedIdentifiers() —Get Associated EIM identifiers" on page 91 —Get Associated EIM Identifiers

Example

See Code disclaimer information for information pertaining to code examples.

The following example will remove an EIM identifier.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;

    EimIdentifierInfo idInfo;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Set identifier information.              */
    idInfo.idtype = EIM_UNIQUE_NAME;
    idInfo.id.uniqueName = "Mary Smith";

    /* Remove this identifier.                  */
    if (0 != (rc = eimRemoveIdentifier(handle,
                                       &idInfo,
                                       err)))
```

```
        printf("Remove identifier error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimRemovePolicyAssociation()—Remove EIM Policy Association

Syntax

```
#include <eim.h>

int eimRemovePolicyAssociation(EimHandle                * eim,
                               EimPolicyAssociationInfo * policyAssoc,
                               EimRC                    * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimRemovePolicyAssociation()** function removes the specified policy association from the domain.

» EIM version 2 must be supported by the local EIM APIs to use this API (see "eimGetVersion()—Get EIM Version" on page 140—Get EIM Version). «

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. LDAP administrators also have access to EIM data. The access groups whose members have authority to the EIM data for this API follow:

- EIM Administrator
- EIM Registries Administrator
- EIM authority to an individual registry

  This authority is needed to the target registry.

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

policyAssoc (Input)

The information about the policy association to be removed.

The EimPolicyAssociationInfo structure contains information about the policy association to remove.

For EIM_CERT_FILTER_POLICY (6) association type, the *policyAssociation* field must contain an EimCertificateFilterPolicyAssociation structure.

For EIM_DEFAULT_REG_POLICY (7) association type, the *policyAssociation* field must contain an EimDefaultRegistryPolicyAssociation structure.

For EIM_DEFAULT_DOMAIN_POLICY (8) association type. the *policyAssociation* field must contain an EimDefaultDomainPolicyAssociation structure.

The structure layouts follow:

```
enum EimAssociationType {
    EIM_ALL_ASSOC,                 /* Not supported on this interface*/
    EIM_TARGET,                    /* Not supported on this interface*/
    EIM_SOURCE,                    /* Not supported on this interface*/
    EIM_SOURCE_AND_TARGET,         /* Not supported on this interface*/
    EIM_ADMIN,                     /* Not supported on this interface*/
    EIM_ALL_POLICY_ASSOC,          /* Not supported on this interface*/
    EIM_CERT_FILTER_POLICY,        /* Association is a certificate
                                      filter policy association.     */
    EIM_DEFAULT_REG_POLICY,        /* Association is a default
                                      registry policy association    */
    EIM_DEFAULT_DOMAIN_POLICY      /* Policy is a default policy for
                                      the domain.                    */
};

typedef struct EimCertificateFilterPolicyAssociation
{
    char  * sourceRegistry;        /* The source registry to remove
                                      the policy association from.   */
    char  * filterValue;           /* The filter value of the policy.*/
    char  * targetRegistry;        /* The name of the target registry
                                      that the filter value is mapped
                                      to.                            */
    char  * targetRegistryUserName; /* The name of the target registry
                                      user name that the filter value
                                      is mapped to.                  */
} EimCertificateFilterPolicyAssociation;

typedef struct EimDefaultRegistryPolicyAssociation
{
    char  * sourceRegistry;        /* The source registry to remove
                                      the policy association from.   */
    char  * targetRegistry;        /* The name of the target registry
                                      that the policy association is
                                      mapped to.                     */
    char  * targetRegistryUserName; /* The name of the target registry
                                      user name that the policy
                                      association is mapped to.      */
} EimDefaultRegistryPolicyAssociation;

typedef struct EimDefaultDomainPolicyAssociation
{
    char  * targetRegistry;        /* The name of the target registry
                                      that the policy association is
                                      mapped to.                     */
    char  * targetRegistryUserName; /* The name of the target registry
                                      user name that the policy
                                      association is mapped to.      */
} EimDefaultDomainPolicyAssociation;

typedef struct EimPolicyAssociationInfo
{
    enum EimAssociationType type;
    union {
        EimCertificateFilterPolicyAssociation   certFilter;
        EimDefaultRegistryPolicyAssociation     defaultRegistry;
        EimDefaultDomainPolicyAssociation       defaultDomain;
    } policyAssociation;
} EimPolicyAssociationInfo;
```

eimrc (Input/Output)
The structure in which to return error code information. If the return value is not 0, eimrc is set with additional information. This parameter may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1)	Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry is not valid or insufficient access to EIM data.

EIMERR_NOREG (28)	EIM Registry not found or insufficient access to EIM data.
EIMERR_NOPOLICYFILTER (61)	Policy filter value not found for the specified EIM Registry.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26)	Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13)	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_PARM_REQ (34)	Missing required parameter. Please check API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_FUNCTION_NOT_SUPPORTED (70)	The specified function is not supported by the EIM version.

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 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.
EIMERR_UNEXP_OBJ_VIOLATION (56) Unexpected object violation.

Related Information

- "eimAddPolicyAssociation()—Add EIM Policy Association" on page 26 —Add EIM Policy Association
- "eimListRegistryAssociations()—List EIM Registry Associations" on page 184 —List EIM Registry Associations

Example

See Code disclaimer information for information pertaining to code examples.

The following example removes a default registry policy association.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;
    EimPolicyAssociationInfo assocInfo;

    /* Set up error structure.                 */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get eim handle from input arg.          */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];
```

```
    /* Set up policy association information    */
    assocInfo.type = EIM_DEFAULT_REG_POLICY;
    assocInfo.policyAssociation.defaultRegistry.sourceRegistry = "MySourceRegistry";
    assocInfo.policyAssociation.defaultRegistry.targetRegistry = "localRegistry";
    assocInfo.policyAssociation.defaultRegistry.targetRegistryUserName = "mjjones";

    /* Remove the policy                        */
    if (0 != (rc = eimRemovePolicyAssociation(handle,
                                              &assocInfo,
                                              err)))
    {
        printf("Remove EIM Policy Association error = %d", rc);
        return -1;
    }

    return 0;

}
```

API introduced: V5R3

Top | Security APIs | APIs by category

---

eimRemovePolicyFilter()—Remove EIM Policy Filter

Syntax

```
#include <eim.h>

int eimRemovePolicyFilter(EimHandle           * eim,
                          EimPolicyFilterInfo * filterInfo,
                          EimRC               * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimRemovePolicyFilter()** function removes the specified policy filter from the domain. When a policy filter is removed, all policy associations to the policy filter are also removed.

» EIM version 2 must be supported by the local EIM APIs to use this API (see "eimGetVersion()—Get EIM Version" on page 140—Get EIM Version). «

Authorities and Locks

EIM Data

Access to EIM data is controlled by EIM access groups. 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

Parameters

eim (Input)

The EIM handle returned by a previous call to eimCreateHandle(). A valid connection is required for this function.

filterInfo (Input)
The information about the policy filter to be removed.

The EimPolicyFilterInfo structure contains information about the policy filter to remove.

For EIM_CERTIFICATE_FILTER (1) policy filter type, the *filter* field must contain an EimCertificatePolicyFilter structure.

The structure layouts follow:

```
enum EimPolicyFilterType {
    EIM_ALL_FILTERS,               /* All policy filters -- not
                                      supported for this interface.  */
    EIM_CERTIFICATE_FILTER         /* Policy filter is a certificate
                                      filter.                        */
};

typedef struct EimCertificatePolicyFilter
{
    char  * sourceRegistry;        /* The source registry to remove the
                                      policy filters from.           */
    char  * filterValue;           /* The policy filter value. A NULL
                                      value will remove all policy
                                      filter values from the registry*/
} EimCertificatePolicyFilter;

typedef struct EimPolicyFilterInfo
{
    enum EimPolicyFilterType type;
    union {
        EimCertificatePolicyFilter   certFilter;
    } filter;
} EimPolicyFilterInfo;
```

eimrc (Input/Output)
The structure in which to return error code information. If the return value is not 0, eimrc is set with additional information. This parameter may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied. Not enough permissions to access data.

EIMERR_ACCESS (1) Insufficient access to EIM data.

EBADDATA
eimrc is not valid.

EBADNAME
Registry is not valid or insufficient access to EIM data.

EIMERR_NOREG (28) EIM Registry not found or insufficient access to EIM data.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) 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 API documentation.
EIMERR_PTR_INVAL (35) Pointer parameter is not valid.
EIMERR_POLICY_FILTER_TYPE_INVAL (60) Policy filter type is not valid.
EIMERR_REGTYPE_INVAL (62) Registry type is not valid.
EIMERR_FUNCTION_NOT_SUPPORTED (70) The specified function is not supported by the EIM version.

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 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.
EIMERR_UNEXP_OBJ_ VIOLATION (56) Unexpected object violation.

Related Information

- "eimAddPolicyFilter()—Add EIM Policy Filter" on page 31 —Add EIM Policy Filter
- "eimListPolicyFilters()—List EIM Policy Filters" on page 166 —List EIM Policy Filters

Example

See Code disclaimer information for information pertaining to code examples.

The following example removes all certificate policy filters for the registry.

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;
    EimPolicyFilterInfo filterInfo;

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up policy information                */
    filterInfo.type = EIM_CERTIFICATE_FILTER;
    filterInfo.filter.certFilter.sourceRegistry = "MySourceRegistry";
    filterInfo.filter.certFilter.filterValue = NULL;

    /* Remove the policy filter                 */
    if (0 != (rc = eimRemovePolicyFilter(handle,
                                         &filterInfo,
                                         err)))
    {
        printf("Remove EIM Policy Filter error = %d", rc);
        return -1;
    }

    return 0;

}
```

API introduced: V5R3

Top | Security APIs | APIs by category

eimRetrieveConfiguration()—Retrieve EIM Configuration

Syntax

```
#include <eim.h>

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

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimRetrieveConfiguration()** function retrieves the EIM configuration information for this system.

Authorities and Locks

No authorization is required.

Parameters

lengthOfEimConfig (Input)
The number of bytes provided by the caller for the configuration information. Minimal size required is 36 bytes.

configData (Output)
A pointer to the data to be returned.

The EimConfig structure contains information about the returned data. The API will return as much data as space has been provided.

EimConfig structure:

```
typedef struct EimConfig
{
    unsigned int bytesReturned;     /* Number of bytes actually returned
                                       by the API.                      */
    unsigned int bytesAvailable;    /* Number of bytes of available data
                                       that could have been returned by
                                       the API.                         */
    int          enable;            /* Flag to indicate if enabled to
                                       participate in EIM domain
                                       0 = not enabled
                                       1 = enabled                      */
    EimListData  ldapURL;           /* ldap URL for domain controller */
    EimListData  localRegistry;     /* Local system registry          */
    EimListData  kerberosRegistry;  /* Kerberos registry              */

    EimListData  x509Registry;      /* X.509 registry                 */

} EimConfig;
```

EimListData structure:

```
typedef struct EimListData
{
    unsigned int length;            /* Length of data                 */
    unsigned int disp;              /* Displacement to data.  This byte
                                       offset is relative to the start of
                                       the parent structure; that is, the
                                       structure containing this
                                       structure.                     */
} EimListData;
```

ccsid (Input)
The ccsid for the output data. If the ccsid is 0 or 65535 the default job ccsid will be used.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EBADDATA
eimrc is not valid.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EINVAL
Input parameter was not valid.

EIMERR_CCSID_INVAL (8) CCSID is outside of valid range or CCSID is not supported.
EIMERR_CONFIG_SIZE (10) Length of EimConfig is not valid.
EIMERR_PARM_REQ (34) Missing required parameter. Please check API documentation.
EIMERR_PTR_INVAL (35) 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.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimSetConfiguration()—Set EIM Configuration" on page 241 —Set EIM Configuration

Example

See Code disclaimer information for information pertaining to code examples.

The following example retrieves the configuration information and prints out the results..

```
#include <eim.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>

void printListData(char * fieldName,
                   void * entry,
                   int offset);
```

```
int main (int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    char          listData[4000];
    EimConfig   * list = (EimConfig * ) listData;

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Get configuration information            */
    if (0 != (rc = eimRetrieveConfiguration(4000,
                                            list,
                                            0,
                                            err)))
    {
        printf("Retrieve configuration error = %d", rc);
        return -1;
    }

    /* Print the results                        */
    printf("____________\n");
    printf("   bytesReturned    = %d\n", list->bytesReturned);
    printf("   bytesAvailable   = %d\n", list->bytesAvailable);
    printf("\n");

    if (0 == list->enable)
        printf("Disabled.\n");
    else
        printf("Enabled.\n");

    printListData("ldap URL",
                  list,
                  offsetof(EimConfig, ldapURL));
    printListData("local Registry",
                  list,
                  offsetof(EimConfig, localRegistry));
    printListData("kerberos registry",
                  list,
                  offsetof(EimConfig, kerberosRegistry));

    printListData("x.509 registry",
                  list,
                  offsetof(EimConfig, x509Registry));


    return 0;
}

void printListData(char * fieldName,
                   void * entry,
                   int offset)
{
    EimListData * listData;
    char * data;
    int dataLength;

    printf("     %s = ",fieldName);
    /* Address the EimListData object */
    listData = (EimListData *)((char *)entry + offset);

    /* Print out results */
    data = (char *)entry + listData->disp;
    dataLength = listData->length;
```

```
    if (dataLength > 0)
        printf("%.*s\n",dataLength, data);
    else
        printf("Not found.\n");

}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimSetAttribute()—Set EIM attributes

Syntax

```
#include <eim.h>

int eimSetAttribute(EimHandle          * eim,
                    enum EimHandleAttr   attrName,
                    void               * attrValue,
                    EimRC              * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimSetAttribute()** function is used to set attributes in the EIM handle structure.

Authorities and Locks

None.

Parameters

eimhandle (Input)
The EIM handle returned by a previous call to eimCreateHandle().

attrName (Input)
The name of the attribute to set. Following are valid values:

EIM_HANDLE_CCSID (0)	This is the CCSID of character data passed by the caller of EIM APIs using the specified EimHandle. 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, eimrc is set with additional information. This parameter may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26)	Unable to allocate internal system object.

EINVAL
Input parameter was not valid.

EIMERR_ATTR_INVAL (5)	Attribute name is not valid.
EIMERR_CCSID_INVAL (8)	CCSID is outside of valid range or CCSID is not supported.
EIMERR_HANDLE_INVAL (17)	EimHandle is not valid.
EIMERR_PARM_REQ (34)	Missing required parameter. Please check API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.

ENOTSUP
Attribute type is not supported.

EIMERR_ATTR_NOTSUPP (6)	Attribute not supported.

EUNKNOWN
Unexpected exception.

EIMERR_UNKNOWN (44)	Unknown error or unknown system state.

Related Information

- "eimCreateHandle()—Create an EIM Handle" on page 70—Create an EIM Handle
- "eimDestroyHandle()—Destroy an EIM Handle" on page 76—Destroy an EIM Handle
- "eimGetAttribute()—Get EIM attributes" on page 98—Get EIM Attributes
- "eimConnectToMaster()—Connect to EIM Master Domain" on page 60—Connect to EIM Master Domain
- "eimConnect()—Connect to EIM Domain" on page 57—Connect to EIM Domain

Example

See Code disclaimer information for information pertaining to code examples.

The following example will set the CCSID attribute in the EIM handle.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimHandle   * handle;
    unsigned int ccsid = 37;

    /* Get eim handle from input arg.           */
    /* This handle is already connected to EIM. */
    handle = (EimHandle *)argv[1];

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Change the CCSID for this handle.        */
    if (0 != (rc = eimSetAttribute(handle,
                                   EIM_HANDLE_CCSID,
                                   (void *)&ccsid,
                                   err)))
        printf("Set Attribute error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimSetConfiguration()—Set EIM Configuration

Syntax

```
#include <eim.h>

int eimSetConfiguration(int             enable,
                        char          * ldapURL,
                        char          * localRegistry,
                        char          * kerberosRegistry,
                        int             ccsid,
      EimRC           * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimSetConfiguration()** function sets the configuration information for use by the system.

Authorities and Locks

The caller of the API must have *SECADM special authority.

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. New connections may not be established with the configured EIM domain
non-zero	Enabled to participate in EIM domain. New connections may be established with the EIM domain.

ldapURL (Input)

A uniform resource locator (URL) that contains the EIM configuration information for the EIM domain controller. This information will be used for all EIM operations. The maximum size for this URL is 1000 bytes.

Possible values are:

NULL	A value of NULL indicates that it 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 the following format:

```
ldap://host:port/dn
      or
ldaps://host:port/dn
```

where:

- `host:port` is the name of the host on which the EIM domain controller is running with an optional port number.
- `dn` is the distinguished name for the domain entry.
- `ldaps` indicates that this host/port combination uses SSL and TLS.

Examples:

- ldap://systemx:389/ibm-eimDomainName=myEimDomain,o=myCompany,c=us
- ldaps://systemy:636/ibm-eimDomainName=thisEimDomain

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 it 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 it 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 the default job ccsid will be used.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied.

EIMERR_AUTH_ERR (7) Insufficient authority for the operation.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EINVAL
Input parameter was not valid.

EIMERR_CCSID_INVAL (8) CCSID is outside of valid range or CCSID is not supported.
EIMERR_CHAR_INVAL (21) A restricted character was used in the object name. Check the API for a list of restricted characters.
EIMERR_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.
EIMERR_INVALID_DN (66) Distinguished Name (DN) is not valid.

ENAMETOOLONG
ldapURL or registry name is too long.

EIMERR_REGNAME_SIZE (39) Local registry name is too large.

EIMERR_URL_SIZE (51) Configuration URL is too large.

ENOMEM
Unable to allocate required space.

EIMERR_NOMEM (27) No memory available. Unable to allocate required space.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimRetrieveConfiguration()—Retrieve EIM Configuration" on page 235 —Retrieve EIM Configuration

Example

See Code disclaimer information for information pertaining to code examples.

The following example sets the configuration information but it is not enabled.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;

    char * ldapURL=
      "ldap://mysystem:389/ibm-eimDomainName=myEIMDomain,o=mycompany,c=us";
    char * local   = "mysystem";
    char * kerberos= "krbprin";

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Set config info, but it is disabled.     */
    if (0 != (rc = eimSetConfiguration(0,
                                       ldapURL,
                                       local,
                                       kerberos,
                                       0,
                                       err)))
        printf("Set configuration error = %d", rc);

    return 0;
}
```

In this example, the configuration information is not changed but it is now enabled for use.

```
#include <eim.h>
#include <stdio.h>

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

```
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Enable configuration info.               */
    if (0 != (rc = eimSetConfiguration(1,
                                       NULL,
                                       NULL,
                                       NULL,
                                       0,
                                       err)))
        printf("Set configuration error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

---

eimSetConfigurationExt()—Set EIM Configuration Extended

Syntax

```
#include <eim.h>

int eimSetConfigurationExt(EimConfigInfo  * configInfo,
                           EimRC          * eimrc)
```

Service Program Name: QSYS/QSYEIM
Default Public Authority: *USE
Threadsafe: Yes

The **eimSetConfigurationExt()** function sets the configuration information for use by the system.

Authorities and Locks

The caller of the API must have *SECADM special authority.

Parameters

configInfo (Input)

The configuration information to be set.

The EimConfigInfo structure contains the configuration information.

For EIM_CONFIG_FORMAT_0 (0) configuration format, the *config* field must contain an EimConfigFormat0 structure.

The structure layouts follow:

```
enum EimConfigFormat {
    EIM_CONFIG_FORMAT_0           /* Information is in configuration
                                     format 0.                     */
};

typedef struct EimConfigFormat0
{
```

```
     char * ldapURL;              /* URL for EIM domain controller. */
     char * localRegistry;        /* Local system registry name.    */
     char * kerberosRegistry;     /* Kerberos registry name.        */
     char * x509Registry;         /* X.509 registry name.           */
} EimConfigFormat0;

typedef struct EimConfigInfo
{
     enum EimConfigFormat format; /* Format of the config info.     */
     int                  enable; /* Indicate if able to establish
                                     new connections in order to
                                     participate in EIM domain
                                     0 = not enabled
                                     1 = enabled                    */
     int                  ccsid;  /* CCSID of input data. If 0 or
                                     65535, default job CCSID will
                                     be used.                       */
     union {
         EimConfigFormat0   format0;
     } config;                    /* Configuration information      */
} EimConfigInfo;
```

Detail description for the configuration information:

- *ldapURL*

  A uniform resource locator (URL) that contains the EIM configuration information for the EIM domain controller. This information will be used for all EIM operations. The maximum size for this URL is 1000 bytes.

  Possible values are:

NULL	A value of NULL indicates that it 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 the following format:

```
ldap://host:port/dn
      or
ldaps://host:port/dn
```

  where:

  – `host:port` is the name of the host on which the EIM domain controller is running with an optional port number.
  – `dn` is the distinguished name for the domain entry.
  – `ldaps` indicates that this host/port combination uses SSL and TLS.

  Examples:

  – ldap://systemx:389/ibm-eimDomainName=myEimDomain,o=myCompany,c=us
  – ldaps://systemy:636/ibm-eimDomainName=thisEimDomain

- *localRegistry*

  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 it 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*

  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 it 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.

- *x509Registry*

  The EIM X.509 registry name. The maximum size for this registry name is 256 bytes.

  Possible values are:

NULL A value of NULL indicates that it should not change.
EIM_CONFIG_NONE (*NONE) This value indicates that there is no X.509 registry for EIM.
registry The EIM X.509 registry name. This is the registry that will be used when adding source associations for user certificates to the EIM identifier.

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the `messageCatalogMessageID` field in the *eimrc* parameter for that value.

0 Request was successful.

EACCES
Access denied.

EIMERR_AUTH_ERR (7) Insufficient authority for the operation.

EBADDATA
eimrc is not valid.

EBUSY
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

ECONVERT
Data conversion error.

EIMERR_DATA_CONVERSION (13) Error occurred when converting data between code pages.

EINVAL
Input parameter was not valid.

EIMERR_CCSID_INVAL (8) CCSID is outside of valid range or CCSID is not supported.
EIMERR_CHAR_INVAL (21) A restricted character was used in the object name. Check the API for a list of restricted characters.
EIMERR_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.
EIMERR_INVALID_DN (66) Distinguished Name (DN) is not valid.
EIMERR_CONFIG_FORMAT_INVAL (68) Configuration format is not valid.

ENAMETOOLONG
ldapURL or registry name is too long.

EIMERR_REGNAME_SIZE (39) Registry name is too large.
EIMERR_URL_SIZE (51) Configuration URL is too large.

ENOMEM
Unable to allocate required space.

EIMERR_NOMEM (27) No memory available. Unable to allocate required space.

EUNKNOWN
Unexpected exception.

EIMERR_LDAP_ERR (23) Unexpected LDAP error. %s
EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "eimRetrieveConfiguration()—Retrieve EIM Configuration" on page 235 —Retrieve EIM Configuration

Example

See Code disclaimer information for information pertaining to code examples.

The following example sets the configuration information but it is not enabled.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
```

```
    EimRC       * err;
    EimConfigInfo configInfo;

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Set up config information.               */
    configInfo.format = EIM_CONFIG_FORMAT_0;
    configInfo.enable = 0;
    configInfo.ccsid = 0;
    configInfo.config.format0.ldapURL =
      "ldap://mysystem:389/ibm-eimDomainName=myEIMDomain,o=mycompany,c=us";
    configInfo.config.format0.localRegistry = "mysystem";
    configInfo.config.format0.kerberosRegistry = "krbprin";
    configInfo.config.format0.x509Registry = "x509reg";

    /* Set config info, but it is disabled.     */
    if (0 != (rc = eimSetConfigurationExt(&.configInfo,
                                          err)))
        printf("Set configuration error = %d", rc);

    return 0;
}
```

In this example, the configuration information is not changed but it is now enabled for use.

```
#include <eim.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int           rc;
    char          eimerr[100];
    EimRC       * err;
    EimConfigInfo configInfo;

    /* Set up error structure.                  */
    memset(eimerr,0x00,100);
    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;

    /* Set up config information.               */
    configInfo.format = EIM_CONFIG_FORMAT_0;
    configInfo.enable = 1;
    configInfo.ccsid = 0;
    configInfo.config.format0.ldapURL = NULL;
    configInfo.config.format0.localRegistry = NULL;
    configInfo.config.format0.kerberosRegistry = NULL;
    configInfo.config.format0.x509Registry = NULL;

    /* Enable configuration info.               */
    if (0 != (rc = eimSetConfigurationExt(&configInfo,
                                          err)))
        printf("Set configuration error = %d", rc);

    return 0;
}
```

API introduced: V5R3

Top | Security APIs | APIs by category

QsySetEIMConnectInfo()—Set EIM Connect Information

Syntax

```
#include <qsyeimapi.h>

#include <eim.h>

int QsySetEIMConnectInfo(enum QsyEimConnectSystem   connectSystem,
       QsyEimConnectionInfo       connectInfo,
                          EimRc                   * eimrc)
```

Service Program Name: QSYS/QSYEIMAPI
Default Public Authority: *USE
Threadsafe: Yes

The **QsySetEIMConnectInfo()** function defines the connection information that will be used by the operating system when it needs to connect to the EIM domain that is configured for this system or for the master system. EIM configuration information is set using eimSetConfiguration().

Authorities and Locks

Authority required
*ALLOBJ and *SECADM special authorities

Parameters

connectSystem
(Input)

The system defined by eimSetConfiguration(). If the configured system is a replica system and EIM updates will be done, then connection information for the master system must also be defined.

QSY_EIM_CONFIG (0)	The specified connection information will be used to connect to the EIM domain that is configured for this system.
QSY_EIM_MASTER (1)	The specified connection information will be used to connect to the master system.

connectInfo
(Input)

The connection information. EIM uses ldap. The connection information indicates the required information to bind to ldap. There are two types of connections supported, simple bind and Kerberos.

If the system is configured to connect to a secure port then Digital Certificate Manager (DCM) must be used to assign a certificate to the Enterprise Identity Mapping Client (QIBM_QSY_EIM_CLIENT) application.

For QSY_EIM_SIMPLE (0) connect type, the *connectInfo* field must contain an EimSimpleConnectInfo structure with a binddn and password. The binddn cannot be longer than 400 bytes. The password cannot be longer than 174 bytes. EimPasswordProtect is used to determine 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 in order to send the protected password.
EIM_PROTECT_CRAM_MD5_OPTIONAL (2)	The protected password will be sent on the bind if the cram-md5 protocol is supported. Otherwise, the ″clear-text″ password is sent.

For QSY_EIM_KERBEROS_KEYTAB (1), connect type, the *connectInfo* field must contain a QsyEimKerberosKeyTab structure with a keytab file name, principal, and realm. Each of the keytab file name, principal, and realm cannot be longer than 400 bytes.

For QSY_EIM_KERBEROS_PWD (2), connect type, the *connectInfo* field must contain a QsyEimKerberosPassword structure with a principal, realm, and password. The principal and realm cannot be longer than 400 bytes. The password cannot be longer than 174 bytes.

For QSY_EIM_REMOVE_CONNECT_INFO (3), connect type, the *connectInfo* field must be zeros. The connection information that is currently defined for the specified connection system will be removed.

Following are the structure layouts:

```
#pragma enumsize(4)

enum QsyEimConnectType {
    QSY_EIM_SIMPLE,
    QSY_EIM_KERBEROS_KEYTAB,
    QSY_EIM_KERBEROS_PWD,
    QSY_EIM_REMOVE_CONNECT_INFO
};

enum EimPasswordProtect {
    EIM_PROTECT_NO,
    EIM_PROTECT_CRAM_MD5,
    EIM_PROTECT_CRAM_MD5_OPTIONAL
};

typedef struct EimSimpleConnectInfo
{
enum EimPasswordProtect  protect;
     char                    reserved[12];
     char                  * bindDn;
     char                  * bindPw;
} EimSimpleConnectInfo;

typedef struct QsyEimKerberosKeyTab
{
     char * keyTabFile;
     char * principal;
     char * realm;
}

typedef struct QsyEimKerberosPassword
{
     char * principal;
     char * realm;
     char * password;
}

typedef struct QsyEimConnectionInfo
{
     enum QsyEimConnectType type;
     union {
EimSimpleConnectInfo   simpleCreds;
         QsyEimKerberosKeyTab   kerberosKeyTab;
         QsyEimKerberosPassword kerberosPassword;
     } connectInfo;
} QsyEimConnectionInfo;
```

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 may be NULL. For the format of the structure, see "EimRC—EIM Return Code" on page 254.

Return Value

The return value from the API. Following each return value is the list of possible values for the messageCatalogMessageID field in the eimrc parameter for that value.

0 Request was successful.

EACCESS (3401)
Access denied. Not enough permissions to set connection information.

EIMERR_AUTH_ERR (7) Insufficient authority for the operation.

EBADDATA (3028)
eimrc is not valid.

EBUSY (3029)
Unable to allocate internal system object.

EIMERR_NOLOCK (26) Unable to allocate internal system object.

EINVAL (3021)
Input parameter was not valid.

EIMERR_PROTECT_INVAL (22)	The protect parameter in EimSimpleConnectInfo is not valid.
EIMERR_PARM_REQ (34)	Missing required parameter. Please check API documentation.
EIMERR_PTR_INVAL (35)	Pointer parameter is not valid.
EIMERR_OS400_CONN_SYS_INVAL (5002)	Connection system is not valid.
EIMERR_RESERVE_INVAL (57)	Reserved field is not valid.

ENAMETOOLONG (3486)
Input parameter is too long.

EIMERR_OS400_BINDDN_SIZE (5001)	Bind DN is too large.
EIMERR_OS400_KEYTAB_SIZE (5003)	Kerberos keytab file name is too large.
EIMERR_OS400_PRINCIPAL_SIZE (5004)	Kerberos principal is too large.
EIMERR_OS400_PWD_SIZE (5005)	Kerberos password is too large.
EIMERR_OS400_REALM_SIZE (5006)	Kerberos realm is too large.

ENOMEM (3460)
Unable to allocate required space.

EIMERR_NOMEM (27) No memory available. Unable to allocate required space.

ENOTSUP (3440)
Connection type is not supported.

EIMERR_CONN_NOTSUPP (12) Connection type is not supported.

EUNKNOWN (3474)
Unexpected exception.

EIMERR_UNKNOWN (44) Unknown error or unknown system state.

Related Information

- "QsyGetEIMConnectInfo()—Get EIM Connect Information" on page 102—Get EIM Connect Information

Example

See Code disclaimer information for information pertaining to code examples.

The following example will set connection information used by the operating system.

```
#include <eim.h>
#include <qsyeimapi.h>

int main(int argc, char *argv[])
{
    int rc;
    enum QsyEimConnectSystem       *connectSys;
    QsyEimConnectionInfo  connectInfo;
    char           eimerr[100];
    EimRC         *err;


    /* Get the system that the connection information is for. */
    connectSys = (enum QsyEimConnectSystem *)argv[1];
    /* Get the type of the connection information. */
    connectInfo.type = *((enum QsyEimConnectType *)argv[2]);
    /* Set the connection information based on the connection type.
    switch (connectInfo.type)      /* Determine connect type.            */
      {
        case QSY_EIM_SIMPLE:
           {
            connectInfo.connectInfo.simpleCreds.protect =
                          *((enum EimPasswordProtect *)argv[3]);
             connectInfo.connectInfo.simpleCreds.bindDn = argv[4];
             connectInfo.connectInfo.simpleCreds.bindPw = argv[5];
            break;
           }
        case QSY_EIM_KERBEROS_KEYTAB:
           {
             connectInfo.connectInfo.kerberosKeyTab.keyTabFile = argv[3];
             connectInfo.connectInfo.kerberosKeyTab.principal = argv[4];
             connectInfo.connectInfo.kerberosKeyTab.realm = argv[5];
             break;
           }
        case QSY_EIM_KERBEROS_PWD:
           {
             connectInfo.connectInfo.kerberosPassword.principal = argv[3];
             connectInfo.connectInfo.kerberosPassword.realm = argv[4];
             connectInfo.connectInfo.kerberosPassword.password = argv[5];
             break;
           }
        case QSY_EIM_REMOVE_CONNECT_INFO:
           {
             connectInfo.connectInfo.kerberosPassword.principal = NULL;
             connectInfo.connectInfo.kerberosPassword.realm = NULL;
             connectInfo.connectInfo.kerberosPassword.password = NULL;
             break;
```

```
        }
      } /* end determine connect type. */

    err = (EimRC *)eimerr;
    err->memoryProvidedByCaller = 100;


    if (0 != (rc = QsySetEIMConnectInfo(*connectSys,
                                        connectInfo,
                                        err)))
        printf("Set connection information error = %d", rc);

    return 0;
}
```

API introduced: V5R2

Top | Security APIs | APIs by category

Concepts

These are the concepts for this category.

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. It can be used 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 (that is, 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 48, or more than 48.

memoryRequiredToReturnData
(Output) The length of the error information available to the API to return, in bytes. 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. This can be used with the messageCatalogID to get the error message text.

messageCatalogMessageID
(Output) The message ID number for the EIM catalog. This can be used with the messageCatalogSetNbr to get the error message text.

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.

Example

See Code disclaimer information for information pertaining to code examples.

The following example shows how to retrieve the message text from the message catalog.

```
#include <nl_types.h>
#include <eim.h>

char  * getError(EimRC * eimrc)
{
    nl_catd catd;
    char * catmsg;
    char * msg = NULL;

    catd = catopen("/QIBM/PRODDATA/OS400/MRI2924/EIM/EIM.CAT", 0);
    if (NULL == catd)
       return NULL;

    catmsg = catgets(catd,
                     eimrc->messageCatalogSetNbr,
                     eimrc->messageCatalogMessageID,
                     strerror(eimrc->returnCode));

    if (catmsg)
    {
       msg = (char *)malloc(strlen(catmsg)+
                        eimrc->substitutionTextLength+1);

       if (0 == eimrc->substitutionTextLength)
          sprintf(msg,catmsg);
       else
          sprintf(msg, catmsg, eimrc->substitutionText);

    }
```

```
    catclose(catd);

    return msg;
}
```

Note: To use the message catalog support in nl_types.h, you must compile the parts with LOCALETYPE(*LOCALE) and SYSIFCOPT(*IFSIO).

"EimRC—EIM Return Code" on page 254 | Security APIs | APIs by category

Appendix. Notices

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

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

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

```
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY  10504-1785
U.S.A.
```

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

```
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan
```

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

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

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

COPYRIGHT LICENSE:

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

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

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

Programming Interface Information

This Application Programming Interfaces (API) publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:
Advanced 36
Advanced Function Printing
Advanced Peer-to-Peer Networking
AFP
AIX
AS/400
COBOL/400
CUA
DB2
DB2 Universal Database
Distributed Relational Database Architecture
Domino
DPI
DRDA
eServer
GDDM
IBM
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
i5/OS
iSeries
Lotus Notes
MVS
Netfinity
Net.Data
NetView
Notes
OfficeVision
Operating System/2
Operating System/400
OS/2
OS/400
PartnerWorld
PowerPC
PrintManager
Print Services Facility
RISC System/6000
RPG/400
RS/6000
SAA
SecureWay
System/36
System/370
System/38
System/390
VisualAge
WebSphere
xSeries

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

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

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

Terms and Conditions

Permissions for the use of these Publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these Publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these Publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these Publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these Publications, or reproduce, distribute or display these Publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the Publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations. IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE

Printed in USA