System i

Programming Validation List APIs

Version 6 Release 1

System i

Programming Validation List APIs

Version 6 Release 1

Note

Before using this information and the product it supports, read the information in “Notices,” on page 53.

This edition applies to version 6, release 1, modification 0 of IBM i5/OS (product number 5761-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

Validation List APIs

Validation lists contain entries that consist of an identifier, data that will be encrypted when it is stored, and free-form data. Entries can be added, changed, removed, found, and validated. You can validate entries by providing the correct entry identifier and data that is encrypted.

One way to use validation lists is to store the user names of a Web browser. The entry identifier would be the user name, the data to encrypt would be the user's password, and the free-form data field would contain any additional data about the user that the browser wanted to store.

The validation list APIs are:

- "Add Validation List Entry (QSYADVLE) API" on page 2 (QSYADVLE) adds an entry to a validation list object.
- "QsyAddValidationLstEntry()—Add Validation List Entry API" on page 6 (QsyAddValidationLstEntry()) adds an entry to a validation list object.
- "Change Validation List Entry (QSYCHVLE) API" on page 11 (QSYCHVLE) changes an entry in a validation list object.
- "QsyChangeValidationLstEntry()—Change Validation List Entry API" on page 16 (QsyChangeValidationLstEntry()) changes an entry in a validation list object.
- "Convert Validation List Entry (QSYCVTVL) API" on page 20 (QSYCVTVL) converts a validation list object from a maximum size of 4 gigabytes to a maximum size of 1 terabyte.
- "QsyFindFirstValidationLstEntry()—Find First Validation List Entry API" on page 21 (QsyFindFirstValidationLstEntry()) finds the first entry in a validation list object and returns information about the validation list entry.
- "QsyFindNextValidationLstEntry()—Find Next Validation List Entry API" on page 24 (QsyFindNextValidationLstEntry()) finds the next entry in a validation list object after the entry that is passed in the Entry_ID parameter and returns information about the validation list entry.
- "Find Validation List Entry (QSYFDVLE) API" on page 27 (QSYFDVLE) finds an entry in a validation list object and returns it.
- "QsyFindValidationLstEntry()—Find Validation List Entry API" on page 32 (QsyFindValidationLstEntry()) finds an entry in a validation list object and returns information about the validation list entry.
- "QsyFindValidationLstEntryAttrs()—Find Validation List Entry Attributes API" on page 36 (QsyFindValidationLstEntryAttrs()) finds an entry in a validation list object, and the attributes associated with the entry.
- "Open List of Validation List Entries (QSYOLVLE) API" on page 41 (QSYOLVLE) returns a list of validation list entries in a validation list object.
- "Remove Validation List Entry (QSYRMVLE) API" on page 48 (QSYRMVLE) removes an entry from a validation list object.
- "QsyRemoveValidationLstEntry()—Remove Validation List Entry API" on page 46 (QsyRemoveValidationLstEntry()) removes an entry from a validation list object.
- "QsyVerifyValidationLstEntry()—Verify Validation List Entry API" on page 49 (QsyVerifyValidationLstEntry()) verifies an entry in a validation list object.

APIs

These are the APIs for this category.

Add Validation List Entry (QSYADVLE) API

Required Parameter Group:

1	Qualified validation list name	Input	Char(20)
2	Entry ID information	Input	Char(*)
3	Data to encrypt information	Input	Char(*)
4	Entry data information	Input	Char(*)
5	Attribute information	Input	Char(*)
6	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The Add Validation List Entry (QSYADVLE) API adds an entry to a validation list object. Entries are stored in hexadecimal sort sequence. The first entry will always be the one in which the entry ID has the smallest hexadecimal value.

Conversions are not done on any data when entries are added. The CCSID value for each field is stored as part of the record but is not used when the entry is added to the validation list.

Authorities and Locks

Validation List Object
*USE and *ADD

Validation List Object Library
*EXECUTE

Required Parameter Group

Qualified validation list name
INPUT; CHAR(20)

The qualified object name of the validation list to add the entry to. The first 10 characters specify the validation list name, and the second 10 characters specify the library.

You can use these special values for the library name:

**CURLIB* The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.
**LIBL* The library list is used to locate the validation list.

Entry ID information
INPUT; CHAR(*)

The format of the entry ID information is as follows. See the "Field Descriptions" on page 4 for more information.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of entry ID
>4	4	BINARY(4)	CCSID of entry ID
>8	8	CHAR(*)	Entry ID

Data to encrypt information
INPUT; CHAR(*)

Data that is associated with the entry ID and is encrypted by the system when it is stored.

The format of the data to encrypt information is as follows. See the "Field Descriptions" on page 4 for more information.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of data to encrypt
4	4	BINARY(4)	CCSID of data to encrypt
8	8	CHAR(*)	Data to encrypt

Entry data information

INPUT; CHAR(*)

Data information that is associated with the entry ID. The format of the entry data information is as follows. See the "Field Descriptions" on page 4 for more information.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of data
4	4	BINARY(4)	CCSID of data
8	8	CHAR(*)	Data

Attribute information

INPUT; CHAR(*)

Attribute information that is associated with the entry. The format of the attribute information is as follows. See the "Field Descriptions" on page 4 for more information.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Number of attributes
4	4	CHAR(*)	Attribute structures

The format of the attribute structure is as follows. See the "Field Descriptions" on page 4 for more information.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of attribute entry
4	4	BINARY(4)	Attribute location
8	8	BINARY(4)	Attribute type
12	C	BINARY(4)	Displacement to attribute ID
16	10	BINARY(4)	Length of attribute ID
20	14	BINARY(4)	Displacement to attribute data
24	18	BINARY(4)	Length of attribute data
		CHAR(*)	Attribute ID
		CHAR(*)	Attribute data

For attributes that are stored in the validation list object, the format of the attribute data is as follows. See the "Field Descriptions" for more information.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	CCSID of attribute
4	4	BINARY(4)	Length of attribute
8	8	CHAR(8)	Reserved
16	10	CHAR(*)	Attribute value

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

Field Descriptions

Attribute data. The information that describes the attribute data.

Attribute ID. The ID of the attribute. For system-defined attributes, the allowed values are:

String value	Description
QsyEncryptData	This is the attribute that is associated with the data to encrypt.

Attribute location. Where the attribute should be stored.

The allowed value is:

0 The attribute is stored in the validation list object.

Attribute structures. Zero or more attribute structures that define the attributes to be associated with the entry.

Attribute type. The type of attribute.

The allowed value follows:

0 This is a system-defined attribute.

Attribute value. The value of the attribute that is associated with the entry.

For the QsyEncryptData attribute, the allowed values follow:

0 The data to be encrypted can only be used to verify an entry. This is the default.

1 The data to be encrypted can be used to verify an entry and can be returned on a find operation. The system value QRETSVRSEC (Retain server security data) is used to determine if the data to be encrypted is stored in the entry or not.

If the system value is set to 0 (Do not retain data), the entry will be added, but the data to be encrypted will not be stored with the entry. The return value from this function will be -2 to indicate that the entry was added, but the data to be encrypted was not stored.

If the system value is set to 1 (Retain data), then the data to be encrypted will be stored in encrypted form when the entry is added.

CCSID of attribute. An integer that represents the CCSID for the attribute. Valid CCSID values are in the range -1 through 65535.

The special values follow:

-1 No CCSID value is stored with the attribute. If the attribute is QsyEncryptData, this value must be specified.
0 The default CCSID for the current user is stored.

CCSID of data to encrypt. An integer that represents the CCSID for the data to encrypt. Valid CCSID values are in the range 1 through 65535.

The special value follows:

0 The default CCSID for the current user is stored.

CCSID of data. An integer that represents the CCSID for the entry data. Valid CCSID values are in the range 1 through 65535.

The special value follows:

0 The default CCSID for the current user is stored.

CCSID of entry ID. An integer that represents the CCSID for the entry ID. Valid CCSID values are in the range 1 through 65535.

The special value follows:

0 The default CCSID for the current user is stored.

Data. The data to store in the validation list entry.

Data to encrypt. The data to be encrypted before storing it in the validation list entry.

Displacement to attribute data. The displacement in the attribute entry to the start of the attribute data information.

Displacement to attribute ID. The displacement in the attribute entry to the start of the attribute ID value.

Entry ID. The data that is used to identify this entry in the validation list.

Length of attribute. The number of bytes of data in the attribute value. The length must be greater than 0. For the QsyEncryptData attribute, the length must be 1.

Length of attribute data. The number of bytes of data in the attribute data structure. The length must be greater than 0.

Length of attribute entry. The length (in bytes) of the current entry. This length can be used to access the next entry, and must be a multiple of 4.

Length of attribute ID. The number of bytes of data in the attribute ID. The length must be greater than 0.

Length of data to encrypt. The number of bytes of data to be encrypted and stored in this validation list entry. Possible values are 0 through 600. If the length is 0, no encrypted data will be stored in the entry.

Length of data. The number of bytes of data to be stored in this validation list entry. Possible values are 0 through 1000. If the length is 0, no data will be stored in the entry.

Length of entry ID. The number of bytes of data that is provided as the entry ID. Possible values are 1 through 100.

Number of attributes. The number of attributes to be added. This value must be greater than or equal to 0. If this value is 0, then no attributes will be added to the entry.

Reserved. This is an ignored field.

Error Messages

Message ID	Error Message Text
CPFA0AA E	Error occurred while attempting to obtain space.
CPF226A E	Validation list entry already exists.
CPF226D E	Not all information stored.
CPF3CF1 E	Error code parameter not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9804 E	Object &2 in library &3 damaged.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R1

Top | Security APIs | APIs by category

QsyAddValidationLstEntry()—Add Validation List Entry API

Syntax

```
#include <qsyvldl.h>

int QsyAddValidationLstEntry
       (Qsy_Qual_Name_T             *Validation_Lst,
        Qsy_Entry_ID_Info_T         *Entry_ID,
        Qsy_Entry_Encr_Data_Info_T  *Encrypt_Data,
        Qsy_Entry_Data_Info_T       *Entry_Data,
        void                        *Attribute_Info);
```

Service Program Name: QSYVLDL
Default Public Authority: *USE
Threadsafe: Yes

The **QsyAddValidationLstEntry()** function adds an entry to a validation list object. Entries are stored in hexadecimal sort sequence. The first entry will always be the one in which the entry ID has the smallest hexadecimal value.

Conversions are not done on any data when entries are added. The CCSID value for each field is stored as part of the record but is not used when the entry is added to the validation list.

Authorities

Validation List Object
*USE and *ADD

Validation List Object Library
*EXECUTE

Parameters

Validation_Lst
(Input) A pointer to the qualified object name of the validation list to add the entry to. The first 10 characters specify the validation list name, and the second 10 characters specify the library. You can use these special values for the library name:

**CURLIB*	The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.
**LIBL*	The library list is used to locate the validation list.

Entry_ID
(Input) A pointer to the entry ID information. The format of the Qsy_Entry_ID_Info_T structure is as follows:

int	Entry_ID_Len	The number of bytes of data that is provided as the entry ID. Possible values are from 1 through 100.
unsigned int	Entry_ID_CCSID	An integer that represents the CCSID for the entry ID. Valid CCSID values are in the range 1 through 65535. The special value follows: *0* The default CCSID for the current user is stored.
unsigned char	Entry_ID[]	The data that is used to identify this entry in the validation list.

Encrypt_Data
(Input) A pointer to data that is associated with the entry ID and is encrypted by the system when it is stored. If the pointer is NULL, there is no encrypted data associated with the entry ID. The format of the Qsy_Entry_Encr_Data_Info_T structure is as follows:

int	Encr_Data_Len	The number of bytes of data to be encrypted and stored in this validation list entry. Possible values are from 1 through 600.
unsigned int	Encr_Data_CCSID	An integer that represents the CCSID for the data to encrypt. Valid CCSID values are in the range 1 through 65535. The special value follows: *0* The default CCSID for the current user is stored.
unsigned char	Encr_Data[]	The data to be encrypted before storing it in the validation list entry.

Entry_Data

(Input) A pointer to the data information that is associated with the entry ID. If the pointer is NULL, there is no data associated with the entry ID. The format of the Qsy_Entry_Data_Info_T structure is as follows:

int	Entry_Data_Len	The number of bytes of data to be stored in this validation list entry. Possible values are from 1 through 1000.
unsigned int	Entry_Data_CCSID	An integer that represents the CCSID for the data. Valid CCSID values are in the range 1 through 65535. The special value follows: *0* The default CCSID for the current user is stored.
unsigned char	Entry_Data[]	The data to be stored in the validation list entry.

Attribute_Info

(Input) A pointer to a structure that contains attribute information that is associated with the entry ID. If the pointer is NULL, there is no attribute information associated with the entry ID. The format of the Qsy_Attr_Info_T structure is as follows:

int	Number_Attrs	The number of attributes being added. This value must be greater than 0.
char	Res_Align[12]	Reserved for boundary alignment.
Qsy_Attr_Descr_T	Attr_Descr[]	An array of attribute description structures.

The format of the Qsy_Attr_Descr_T structure is as follows:

int	Attr_Location	Where the attribute should be stored. The allowed value follows: *0 QSY_IN_VLDL* The attribute is stored in the validation list object.
int	Attr_Type	The type of attribute. The allowed value follows: *0 QSY_SYSTEM_ATTR* This is a system-defined attribute.
union	Attr_Res Res_1[8]	Reserved data. This value must be hexadecimal zero.
char *	Attr_ID	The ID of the attribute. For system-defined attributes, the allowed value is: *String value* Description *QsyEncryptData* This is the attribute that is associated with the data to encrypt.
union	Attr_Other_Descr Res_1[32]	Reserved data. This value must be hexadecimal zero.
union	Attr_Data_Info	The information describing the attribute data.
union	Attr_Other_Data Res_1[32]	Reserved data. This value must be hexadecimal zero.

The format of the Attr_Data_Info union is as follows:

Qsy_In_VLDL_T	Attr_VLDL	The attribute data information for an attribute that is stored in the validation list object.
union	Attr_In_Other Res_1[96]	Reserved data. The last 64 bytes must be zero.

The format of the Qsy_In_VLDL_T structure is as follows:

int	Attr_CCSID	An integer that represents the CCSID for the attribute. Valid CCSID values are in the range -1 through 65535. The special values follow: *-1* No CCSID value is stored with the attribute. If the attribute is QsyEncryptData, this value must be specified. *0* The default CCSID for the current user is stored.
int	Attr_Len	The number of bytes of data in the attribute value. The length must be greater than 0. For the QsyEncryptData attribute, the length must be 1.
union	Attr_Res Res_1[8]	Reserved data. This value must be hexadecimal zero.
void *	Attr_Value	A pointer to the value of the attribute associated with the entry. For the QsyEncryptData attribute, the allowed values follow: *0 QSY_VFY_ONLY* The data to be encrypted can only be used to verify an entry. This is the default. *1 QSY_VFY_FIND* The data to be encrypted can be used to verify an entry and can be returned on a find operation.

If the QSY_VFY_FIND value is specified for the QsyEncryptData attribute, the system value QRETSVRSEC (Retain server security data) is used to determine if the data to be encrypted is stored in the entry or not.

If the system value is set to 0 (Do not retain data), the entry will be added, but the data to be encrypted will not be stored with the entry. The return value from this function will be -2 to indicate that the entry was added, but the data to be encrypted was not stored.

If the system value is set to 1 (Retain data), then the data to be encrypted will be stored when the entry is added.

Return Value

0 **QsyAddValidationLstEntry()** was successful.
-1 **QsyAddValidationLstEntry()** was not successful. The *errno* global variable is set to indicate the error.
-2 **QsyAddValidationLstEntry()** was successful, but the data to be encrypted was not stored.

Error Conditions

If **QsyAddValidationLstEntry()** is not successful, *errno* indicates one of the following errors:

3401 [EACCES]

The current user does not have *USE and *ADD authorities to the validation list object, or does not have *EXECUTE authority to the validation list object library.

3406 [EAGAIN]

The validation list object is currently locked by another process.

3484 [EDAMAGE]

The validation list object is damaged.

3457 [EEXIST]

Specified entry already exists.

3021 [EINVAL]

Parameter value is not valid.

3025 [ENOENT]

The validation list object was not found.

3404 [ENOSPC]

No space available.

3474 [EUNKNOWN]

Unknown system state. Check the job log for a CPF9872 message.

Example

The following example adds an entry for a user named FRED to the validation list object WEBUSRS. FRED has encrypted data (password), but no other data. The CCSID for the entry ID is set to the current user's default CCSID. The CCSID for the encryption data is set to 65535.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 52.

```
#include <qsyvldl.h>

main()
{
  #define VLD_LST  "WEBUSRS   WEBLIB    "
  Qsy_Entry_ID_Info_T    entry_info;
  Qsy_Entry_Encr_Data_Info_T  encrypt_data;

  entry_info.Entry_ID_Len = 4;
  entry_info.Entry_ID_CCSID = 0;
  strncpy(entry_info.Entry_ID,"FRED",entry_info.Entry_ID_Len);
  encrypt_data.Encr_Data_Len = 7;
  strncpy(encrypt_data.Encr_Data,"N1LJDTS",
            encrypt_data.Encr_Data_Len);
  encrypt_data.Encr_Data_CCSID = 65535;

  if (0 != QsyAddValidationLstEntry((Qsy_Qual_Name_T *)&VLD_LST,
                                    &entry_info,
                                    &encrypt_data,
                                    NULL,
                                    NULL))
     perror("QsyAddValidationLstEntry()");

}
```

API introduced: V4R1

Change Validation List Entry (QSYCHVLE) API

Required Parameter Group:

1	Qualified validation list name	Input	Char(20)
2	Entry ID information	Input	Char(*)
3	Data to encrypt information	Input	Char(*)
4	Entry data information	Input	Char(*)
5	Attribute information	Input	Char(*)
6	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The Change Validation List Entry (QSYCHVLE) API changes an entry in a validation list object. The data to be encrypted, the entry data values, and some of the entry attributes may be changed.

To identify an entry to be changed, there must be an exact match in the entry for the value that is specified in the entry ID parameter and the length of the entry ID. For example, an entry ID value of SMITH with a length of 5 would not allow you to change an entry where the entry ID is SMITH and the length is 7.

Conversions are not done on any data when entries are changed. The CCSID values for the fields are stored as part of the record but are not used when the entry is changed.

Authorities and Locks

Validation List Object
*USE and *UPD

Validation List Object Library
*EXECUTE

Required Parameter Group

Qualified validation list name
INPUT; CHAR(20)

The qualified object name of the validation list that contains the entry to change. The first 10 characters specify the validation list name, and the second 10 characters specify the library.

You can use these special values for the library name:

**CURLIB*	The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.
**LIBL*	The library list is used to locate the validation list.

Entry ID information
INPUT; CHAR(*)

The format of the entry ID information is as follows. See the "Field Descriptions" on page 13 for more information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of entry ID
4	4	BINARY(4)	CCSID of entry ID
8	8	CHAR(*)	Entry ID

Data to encrypt information
INPUT; CHAR(*)

The data is encrypted by the system when it is stored. The format of the data to encrypt information is as follows. See the "Field Descriptions" on page 13 for more information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of data to encrypt
4	4	BINARY(4)	CCSID of data to encrypt
8	8	CHAR(*)	Data to encrypt

Entry data information
INPUT; CHAR(*)

The format of the entry data information is as follows. See the "Field Descriptions" on page 13 for more information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of data to encrypt
0	0	BINARY(4)	Length of data
4	4	BINARY(4)	CCSID of data
8	8	CHAR(*)	Data

Attribute information
INPUT; CHAR(*)

Attribute information that is associated with the entry. The format of the attribute information is as follows. See the "Field Descriptions" on page 13 for more information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of data to encrypt
0	0	BINARY(4)	Number of attributes
4	4	CHAR(*)	Attribute structures

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of attribute entry
4	4	BINARY(4)	Attribute location

Offset		Type	Field
Dec	Hex		
8	8	BINARY(4)	Attribute type
12	C	BINARY(4)	Displacement to attribute ID
16	10	BINARY(4)	Length of attribute ID
20	14	BINARY(4)	Displacement to attribute data
24	18	BINARY(4)	Length of attribute data
		CHAR(*)	Attribute ID
		CHAR(*)	Attribute data

For attributes that are stored in the validation list object, the format of the attribute data is as follows. See the "Field Descriptions" for more information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of data to encrypt
0	0	BINARY(4)	CCSID of attribute
4	4	BINARY(4)	Length of attribute
8	8	CHAR(8)	Reserved
16	10	CHAR(*)	Attribute value

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

Field Descriptions

Attribute data. The information that describes the attribute data.

Attribute ID. The ID of the attribute.

For system-defined attributes, the allowed value is:

String value	Description
QsyEncryptData	This is the attribute that is associated with the data to encrypt. This attribute can only be changed if the length of data to encrypt is not -1.

Attribute location. Where the attribute should be stored.

The allowed value is:

0 The attribute is stored in the validation list object.

Attribute structures. Zero or more attribute structures that define the attributes associated with the entry.

Attribute type. The type of attribute.

The allowed value follows:

0 This is a system-defined attribute.

Attribute value. The value of the attribute that is associated with the entry.

For the QsyEncryptData attribute, the allowed values follow:

0 The data to be encrypted can only be used to verify an entry. This is the default.
1 The data to be encrypted can be used to verify an entry and can be returned on a find operation. The system value QRETSVRSEC (Retain server security data) is used to determine if the data to be encrypted is stored in the entry or not.

If the system value is set to 0 (Do not retain data), the entry will be added, but the data to be encrypted will not be stored with the entry. The return value from this function will be -2 to indicate that the entry was added, but the data to be encrypted was not stored.

If the system value is set to 1 (Retain data), then the data to be encrypted will be stored in encrypted form when the entry is added.

CCSID of attribute. An integer that represents the CCSID for the attribute. Valid CCSID values are in the range -1 through 65535.

The special values follow:

-1 No CCSID value is stored with the attribute. If the attribute is QsyEncryptData, this value must be specified.
0 The default CCSID for the current user is stored.

CCSID of data to encrypt. An integer that represents the CCSID for the data to encrypt. Valid CCSID values are in the range 1 through 65535.

The special value follows:

0 The default CCSID for the current user is stored.

CCSID of data. An integer that represents the CCSID for the entry data. Valid CCSID values are in the range 1 through 65535.

The special value follows:

0 The default CCSID for the current user is stored.

CCSID of entry ID. An integer that represents the CCSID for the entry ID. Valid CCSID values are in the range 0 through 65535. This field is not used to change the entry.

Data. The data to store in the validation list entry.

Data to encrypt. The data to be encrypted before storing it in the validation list entry.

Displacement to attribute data. The displacement in the attribute entry to the start of the attribute data information.

Displacement to attribute ID. The displacement in the attribute entry to the start of the attribute ID value.

Entry ID. The data that is used to identify this entry in the validation list.

Length of attribute. The number of bytes of data in the attribute value. The length must be greater than or equal to 0. If a length of 0 is specified, the attribute is removed from the entry. For the QsyEncryptData attribute, the maximum length is 1.

Length of attribute data. The number of bytes of data in the attribute data structure. The length must be greater than 0.

Length of attribute entry. The length (in bytes) of the current entry. This length can be used to access the next entry, and must be a multiple of 4.

Length of attribute ID. The number of bytes of data in the attribute ID. The length must be greater than 0.

Length of data to encrypt. The number of bytes of data to be encrypted and stored in this validation list entry. Possible values are -1 through 600. If the length is 0, any encrypted data that is associated with the entry ID will be removed. If the length is -1, the encrypted data that is associated with the entry ID is not changed.

Length of data. The number of bytes of data to be stored in this validation list entry. Possible values are -1 through 1000. If the length is 0, any data that is associated with the entry ID will be removed. If the length is -1, the data that is associated with the entry ID is not changed.

Length of entry ID. The number of bytes of data that is provided as the entry ID. Possible values are 1 through 100.

Number of attributes. The number of attributes to be added. This value must be greater than or equal to 0. If this value is 0, then no attributes will be changed in the entry.

Reserved. This is an ignored field.

Error Messages

Message ID	Error Message Text
CPFA0AA E	Error occurred while attempting to obtain space.
CPF226B E	Validation list entry does not exist.
CPF226D E	Not all information stored.
CPF3CF1 E	Error code parameter not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9804 E	Object &2 in library &3 damaged.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R2

Top | Security APIs | APIs by category

QsyChangeValidationLstEntry()—Change Validation List Entry API

Syntax

```
#include <qsyvldl.h>

int QsyChangeValidationLstEntry
       (Qsy_Qual_Name_T             *Validation_Lst,
        Qsy_Entry_ID_Info_T         *Entry_ID,
        Qsy_Entry_Encr_Data_Info_T  *Encrypt_Data,
        Qsy_Entry_Data_Info_T       *Entry_Data,
        void                        *Attribute_Info);
```

Service Program Name: QSYVLDL
Default Public Authority: *USE
Threadsafe: Yes

The **QsyChangeValidationLstEntry()** function changes an entry in a validation list object. The data to be encrypted, the entry data values, and some of the entry attributes may be changed.

To identify an entry to be changed, there must be an exact match in the entry for the value that is specified in the *Entry_ID* parameter and the length of the entry ID. For example, an entry ID value of ″SMITH″ with a length of 5 would not allow you to change an entry where the entry ID is ″SMITH″ and the length is 7.

Conversions are not done on any data when entries are changed. The CCSID values are stored as part of the record, to be available to the user of the API, but are not used when the entry is changed.

Authorities

Validation List Object
*USE and *UPD

Validation List Object Library
*EXECUTE

Parameters

Validation_Lst
(Input)

A pointer to the qualified object name of the validation list that contains the entry to change. The first 10 characters specify the validation list name, and the second 10 characters specify the library. You can use these special values for the library name:

*CURLIB	The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.
*LIBL	The library list is used to locate the validation list.

Entry_ID
(Input)

A pointer to the entry ID information. The Qsy_Entry_ID_Info_T structure is as follows:

int	Entry_ID_Len	The number of bytes of data that is provided as the entry ID. Possible values are from 1 through 100.
unsigned int	Entry_ID_CCSID	An integer that represents the CCSID for the entry ID. Valid CCSID values are in the range 0 through 65535. This field is not used to change the entry.

unsigned char	Entry_ID[]	The data that is used to identify this entry in the validation list.

Encrypt_Data
(Input)

A pointer to the data that is associated with the entry ID. The data is encrypted by the system when it is stored. If the pointer is NULL, the encrypted data that is associated with the entry ID is not changed. The format of the Qsy_Entry_Encr_Data_Info_T structure is as follows:

int	Encr_Data_Len	The number of bytes of data to be encrypted and stored in this validation list entry. Possible values are from 0 through 600.
unsigned int	Encr_Data_CCSID	An integer that represents the CCSID for the data to encrypt. Valid CCSID values are in the range 1 through 65535. The special value follows: *0* The default CCSID for the current user is stored.
unsigned char	Encr_Data[]	The data to be encrypted before storing it in the validation list entry.

If Encr_Data_Len is 0, any encrypted data that is associated with the entry ID will be removed.

Entry_Data
(Input)

A pointer to the data information that is associated with the entry ID. If the pointer is NULL, the data that is associated with the entry ID is not changed. The format of the Qsy_Entry_Data_Info_T structure is as follows:

int	Entry_Data_Len	The number of bytes of data to be stored in this validation list entry. Possible values are from 0 through 1000.
unsigned int	Entry_Data_CCSID	An integer that represents the CCSID for the data. Valid CCSID values are in the range 1 through 65535. The special value follows: *0* The default CCSID for the current user is stored.
unsigned char	Entry_Data[]	The data to be stored in the validation list entry.

If the Entry_Data_Length is 0, any data that is associated with the entry ID will be removed.

Attribute_Info
(Input)

A pointer to a structure that contains attribute information that is associated with the entry ID. If the pointer is NULL, the attributes associated with the entry ID are not changed. The format of the Qsy_Attr_Info_T structure is as follows:

int	Number_Attrs	The number of attributes being changed. This value must be greater than 0.
char	Res_Align[12]	Reserved for boundary alignment.
Qsy_Attr_Descr_T	Attr_Descr[]	An array of attribute description structures.

The format of the Qsy_Attr_Descr_T structure is as follows:

int	Attr_Location	Where the attribute should be stored. The allowed value follows: *0 QSY_IN_VLDL* The attribute is stored in the validation list object.

int	Attr_Type	The type of attribute. The allowed value follows: *0 QSY_SYSTEM_ATTR* This is a system-defined attribute.
union	Attr_Res Res_1[8]	Reserved data. This value must be hexadecimal zero.
char *	Attr_ID	The ID of the attribute. For system-defined attributes, the allowed value is:
	String value	Description
	QsyEncryptData	This is the attribute that is associated with the data to encrypt. This attribute can only be changed if the Encrypt_Data parameter is not NULL.
union	Attr_Other_Descr Res_1[32]	Reserved data. This value must be hexadecimal zero.
union	Attr_Data_Info	The information that describes the attribute data.
union	Attr_Other_Data Res_1[32]	Reserved data. This value must be hexadecimal zero.

The format of the Attr_Data_Info_T union is as follows:

Qsy_In_VLDL_T	Attr_VLDL	The attribute data information for an attribute that is stored in the validation list object.
union	Attr_In_Other Res_1[96]	Reserved data. The last 64 bytes must be zero.

The format of the Qsy_In_VLDL_T structure is as follows:

int	Attr_CCSID	An integer that represents the CCSID for the attribute. Valid CCSID values are in the range -1 through 65535. The special values follow: *-1* No CCSID value is stored with the attribute. If the attribute is QsyEncryptData, this value is assumed. *0* The default CCSID for the current user is stored.
int	Attr_Len	The number of bytes of data in the attribute value. The length must be greater than or equal to 0. If a length of 0 is specified, the attribute is removed from the entry. For the QsyEncryptData attribute, the maximum length is 1.
union	Attr_Res Res_1[8]	Reserved data. This value must be hexadecimal zero.
void *	Attr_Value	Pointer to the value of the attribute associated with the entry. For the QsyEncryptData attribute, the allowed values follow: *0 QSY_VFY_ONLY* The data to be encrypted can only be used to verify an entry. This is the default. *1 QSY_VFY_FIND* The data to be encrypted can be used to verify an entry and can be returned on a find operation.

If the QSY_VFY_FIND value is specified for the QsyEncryptData attribute, the system value QRETSVRSEC (Retain server security data) is used to determine if the data to be encrypted is stored in the entry or not. If the system value is set to 0 (Do not retain data), the entry will be changed, but the data to be encrypted will not be stored with the entry. The return value from

this function will be -2, to indicate that the entry was changed, but the data to be encrypted was not stored. If the system value is set to 1 (Retain data), then the data to be encrypted will be stored when the entry is changed.

Return Value

0 **QsyChangeValidationLstEntry()** was successful.
-1 **QsyChangeValidationLstEntry()** was not successful. The *errno* global variable is set to indicate the error.
-2 **QsyChangeValidationLstEntry()** was successful, but the data to be encrypted was not stored.

Error Conditions

If **QsyChangeValidationLstEntry()** is not successful, *errno* indicates one of the following errors.

3401 [EACCES]

The current user does not have *USE and *UPD authorities to the validation list object, or does not have *EXECUTE authority to the validation list object library.

3406 [EAGAIN]

The validation list object is currently locked by another process.

3484 [EDAMAGE]

The validation list object is damaged.

3021 [EINVAL]

Parameter value is not valid.

3025 [ENOENT]

The validation list object was not found.

3026 [ENOREC]

Specified entry does not exist.

3404 [ENOSPC]

No space available.

3474 [EUNKNOWN]

Unknown system state. Check the job log for a CPF9872 message.

Example

The following example changes an entry for a user named FRED in the validation list object WEBUSRS. FRED's encrypted data (password) and the CCSID for the encrypted data are being changed, but not any other data.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 52.

```
#include <qsyvldl.h>

main()
{
  #define VLD_LST  "WEBUSRS   WEBLIB    "
  Qsy_Entry_ID_Info_T    entry_info;
  Qsy_Entry_Encr_Data_Info_T  encrypt_data;

  entry_info.Entry_ID_Len = 4;
  strncpy(entry_info.Entry_ID,"FRED",entry_info.Entry_ID_Len);
  encrypt_data.Encr_Data_Len = 7;
  encrypt_data.Encr_Data_CCSID = 37;
```

```
  strncpy(encrypt_data.Encr_Data,"MSN1TJG",
          encrypt_data.Encr_Data_Len);

  if (0 != QsyChangeValidationLstEntry(
                        (Qsy_Qual_Name_T *)&VLD_LST,
                        &entry_info,
                        &encrypt_data,
                        NULL,
                        NULL))
     perror("QsyChangeValidationLstEntry()");

}
```

API introduced: V4R1

Top | Security APIs | APIs by category

Convert Validation List Entry (QSYCVTVL) API

Required Parameter Group:

1	Qualified validation list name	Input	Char(20)
2	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: Yes

The Convert Validation List (QSYCVTVL) API converts a validation list object from a maximum size of 4 gigabytes to a maximum size of 1 terabyte. Converting a validation list to a 1 terabyte validation list will allow for more entries to be stored in the validation list. Also, the existing entries are stored more efficiently in a 1 terabyte validation list.

Note: If the validation list is converted to a 1 terabyte validation list, it cannot be saved to a release prior to Version 5 Release 2 Modification 0.

Note: The validation list to be converted must reside in the current library namespace.

Authorities and Locks

Validation List Object Authority
*OBJMGT

*OBJEXIST

Validation List Library Authority
*EXECUTE

*ADD

Validation List Object Lock
*EXCL

Validation List Library Lock
*SHRUPD

Required Parameter Group

Qualified validation list name
INPUT; CHAR(20)

The qualified object name of the validation list to be converted. The first 10 characters specify the validation list name, and the second 10 characters specify the library.

You can use these special values for the library name:

**CURLIB* The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.

**LIBL* The library list is used to locate the validation list.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

Error Messages

Message ID	Error Message Text
CPF2113 E	Cannot allocate library &1.
CPF2122 E	Storage limit exceeded for user profile &1.
CPF3CF1 E	Error code parameter not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9804 E	Object &2 in library &3 damaged.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R4

Top | Security APIs | APIs by category

QsyFindFirstValidationLstEntry()—Find First Validation List Entry API

Syntax

```
#include <qsyvldl.h>

int QsyFindFirstValidationLstEntry
       (Qsy_Qual_Name_T        *Validation_Lst,
        Qsy_Rtn_Vld_Lst_Ent_T  *First_Entry);
```

Service Program Name: QSYVLDL
Default Public Authority: *USE
Threadsafe: Yes

The **QsyFindFirstValidationLstEntry()** function finds the first entry in a validation list object. The function then returns the information for the first entry in the buffer that is pointed to by the *First_Entry* parameter. The entries are stored in hexadecimal sort sequence, so the first entry will be the one where the entry ID has the smallest hexadecimal value.

Authorities

Validation List Object

*USE

Validation List Object Library
*EXECUTE

Note: If the QsyEncryptData attribute is set to QSY_VFY_FIND_E (1), then the user must have *USE, *ADD, and *UPD authority to the validation list to get the data to be encrypted returned in the *First_Entry* parameter.

Parameters

Validation_Lst
(Input)

A pointer to the qualified object name of the validation list to find the first entry in. The first 10 characters specify the validation list name, and the second 10 characters specify the library. You can use these special values for the library name:

**CURLIB*	The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.
**LIBL*	The library list is used to locate the validation list.

First_Entry
(Output)

A pointer to the buffer where the first entry information is placed. The buffer must be allocated to the size of the *Qsy_Rtn_Vld_Lst_Ent_T* structure or the results will be unpredictable.

The format of the *Qsy_Rtn_Vld_Lst_Ent_T* structure is as follows:

Qsy_Entry_ID_Info_T	**Entry_ID_Info**	**The entry ID information structure.**
Qsy_Entry_Encr_Data_Info_T	Encr_Data_Info	The data to be encrypted information structure.
Qsy_Entry_Data_Info_T	Entry_Data_Info	The entry data information structure.
char	Reserved[4]	This is an ignored field.
void *	Entry_More_Info	A pointer to additional information. This pointer is currently set to NULL.

The format of the Qsy_Entry_ID_Info_T structure is as follows:

int	Entry_ID_Len	The length of the entry ID.
unsigned int	Entry_ID_CCSID	The CCSID associated with the entry ID.
unsigned char	Entry_ID[100]	The entry ID.

The format of the Qsy_Entry_Encr_Data_Info_T structure is as follows:

int	Encr_Data_Len	The number of bytes of encrypted data that is stored in this validation list entry. If the QsyEncryptData attribute is 0 or the QRETSVRSEC system value is '0', the length will always be 0.
unsigned int	Encr_Data_CCSID	The CCSID associated with the encrypted data.
unsigned char	Encr_Data[600]	If the QsyEncryptData attribute is 1 and the QRETSVRSEC system value is '1', then the encrypted data that is stored in the entry will be decrypted and returned in this field. If the QsyEncryptData attribute is 0 or the QRETSVRSEC system value is '0', then the encrypted data cannot be returned, and the contents of this field are unpredictable.

The format of the Qsy_Entry_Data_Info_T structure is as follows:

int	Entry_Data_Len	The length of the entry data.
unsigned int	Entry_Data_CCSID	The CCSID associated with the entry data.
unsigned char	Entry_Data[1000]	The entry data.

Return Value

0 QsyFindFirstValidationLstEntry() was successful. The return value points to the entry.
-1 QsyFindFirstValidationLstEntry() was not successful. The *errno* global variable is set to indicate the error.

Error Conditions

If **QsyFindFirstValidationLstEntry()** is not successful, *errno* indicates one of the following errors:

3401 [EACCES]

The current user does not have *USE authority to the validation list object, or does not have *EXECUTE authority to the validation list object library.

3406 [EAGAIN]

The validation list object is currently locked by another process.

3484 [EDAMAGE]

The validation list object is damaged.

3021 [EINVAL]

Parameter value is not valid.

3025 [ENOENT]

The validation list object was not found.

3026 [ENOREC]

There are no entries in the validation list object.

3474 [EUNKNOWN]

Unknown system state. Check the job log for a CPF9872 message.

Example

The following example finds all the entries in the validation list object WEBUSRS.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 52.

```
#include <qsyvldl.h>
#include <errno.h>

main()
{
  #define VLD_LST  "WEBUSRS   WEBLIB    "
  Qsy_Rtn_Vld_Lst_Ent_T  entry_1;
  Qsy_Rtn_Vld_Lst_Ent_T  entry_2;
  Qsy_Rtn_Vld_Lst_Ent_T  *input_info,
                         *output_info,
                         *temp;
  Qsy_Entry_ID_Info_T    *input_entry;
  short int              i;
  int                    rtn_errno;
```

```
  /* Initialize pointers to input and output buffers.              */
  output_info = addr(entry_1);
  input_info = addr(entry_2);
  /* Get the first entry in the validation list.                   */
  rtn_errno = QsyFindFirstValidationLstEntry(
                            (Qsy_Qual_Name_T *)&VLD_LST,
                            output_info))

  while (0 == rtn_errno)
  { /* Process all the entries in the validation list.            */
          .
          .
          .
     (process the entry)
          .
          .
          .
    /* Switch the pointers to the buffers so that the output from  */
    /* the last find operation is used as input to the 'find-next'
    /* operation.                                                  */
    temp = output_info;
    output_info = input_info;
    input_info = temp;

    /* Find the next entry.                                        */
    rtn_errno = QsyFindNextValidationLstEntry(
                             (Qsy_Qual_Name_T *)&VLD_LST,
                             &input_info->Entry_ID_Info,
                             output_info))
  }
  /* Check if an error occurred.                                   */
  if (0 != rtn_errno && ENOREC != errno)
    perror("Find of validation list entry");

}
```

API introduced: V4R1

Top | Security APIs | APIs by category

QsyFindNextValidationLstEntry()—Find Next Validation List Entry API

Syntax

```
#include <qsyvldl.h>

int QsyFindNextValidationLstEntry
      (Qsy_Qual_Name_T        *Validation_Lst,
       Qsy_Entry_ID_Info_T    *Entry_ID,
       Qsy_Rtn_Vld_Lst_Ent_T  *Next_Entry);
```

Service Program Name: QSYVLDL
Default Public Authority: *USE
Threadsafe: Yes

The **QsyFindNextValidationLstEntry()** function finds the next entry in a validation list object after the entry that is passed in the *Entry_ID* parameter. It then returns the information for the next entry in the buffer that is pointed to by the *Next_Entry* parameter. The entries are stored in hexadecimal sort sequence; therefore, the next entry will be the one with an entry ID whose hexadecimal value would follow the hexadecimal value of the entry passed in the *Entry_ID* parameter. The entry specified in the

Entry_ID parameter does not need to exist in the validation list, and this function does not have to follow a **QsyFindFirstValidationLstEntry()** or **QsyFindValidationLstEntry()** function call.

Authorities

Validation List Object
*USE

Validation List Object Library
*EXECUTE

Note: If the QsyEncryptData attribute is set to QSY_VFY_FIND_E (1), then the user must have *USE, *ADD, and *UPD authority to the validation list to get the data to be encrypted returned in the *Next_Entry* parameter.

Parameters

Validation_Lst
(Input)

A pointer to the qualified object name of the validation list to find the next entry in. The first 10 characters specify the validation list name, and the second 10 characters specify the library. You can use these special values for the library name:

**CURLIB*	The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.
**LIBL*	The library list is used to locate the validation list.

Entry_ID
(Input)

A pointer to the entry ID information. The format of the Qsy_Entry_ID_Info_T structure is as follows:

int	Entry_ID_Len	The number of bytes of data that is provided as the entry ID. Possible values are from 1 through 100.
unsigned int	Entry_ID_CCSID	An integer that represents the CCSID for the entry ID. Valid CCSID values are in the range 0 through 65535. This value is not used to find the entry.
unsigned char	Entry_ID[100]	The data that is used to identify this entry in the validation list.

Next_Entry
(Output)

A pointer to the buffer where the next entry information is placed. The buffer must be allocated to the size of the *Qsy_Rtn_Vld_Lst_Ent_T* structure or the results will be unpredictable. The format of the *Qsy_Rtn_Vld_Lst_Ent_T* structure is as follows:

Qsy_Entry_ID_Info_T	Entry_ID_Info	The entry ID information structure.
Qsy_Entry_Encr_Data_Info_T	Encr_Data_Info	The data to be encrypted information structure.
Qsy_Entry_Data_Info_T	Entry_Data_Info	The entry data information structure.
char	Reserved[4]	This is an ignored field.
void *	Entry_More_Info	A pointer to additional information. This pointer is currently set to NULL.

See the Entry_ID (page 25) parameter for the format of the Qsy_Entry_ID_Info_T structure.

The format of the Qsy_Entry_Encr_Data_Info_T structure is as follows:

int	Encr_Data_Len	The number of bytes of encrypted data that is stored in this validation list entry. If the QsyEncryptData attribute is 0 or the QRETSVRSEC system value is '0', the length will always be 0.
unsigned int	Encr_Data_CCSID	The CCSID associated with the encrypted data.
unsigned char	Encr_Data[600]	If the QsyEncryptData attribute is 1 and the QRETSVRSEC system value is '1', then the encrypted data that is stored in the entry will be decrypted and returned in this field. If the QsyEncryptData attribute is 0 or the QRETSVRSEC system value is '0', then the encrypted data cannot be returned, and the contents of this field are unpredictable.

The format of the Qsy_Entry_Data_Info_T structure is as follows:

int	Entry_Data_Len	The length of the entry data.
unsigned int	Entry_Data_CCSID	The CCSID associated with the entry data.
unsigned char	Entry_Data[1000]	The entry data.

Return Value

0 **QsyFindNextValidationLstEntry()** was successful. The return value points to the entry.
-1 **QsyFindNextValidationLstEntry()** was not successful. The *errno* global variable is set to indicate the error.

Error Conditions

If **QsyFindNextValidationLstEntry()** is not successful, *errno* indicates one of the following errors:

3401 [EACCES]

The current user does not have *USE authority to the validation list object, or does not have *EXECUTE authority to the validation list object library.

3406 [EAGAIN]

The validation list object is currently locked by another process.

3484 [EDAMAGE]

The validation list object is damaged.

3021 [EINVAL]

Parameter value is not valid.

3025 [ENOENT]

The validation list object was not found.

3026 [ENOREC]

There are no more entries in the validation list object.

3474 [EUNKNOWN]

Unknown system state. Check the job log for a CPF9872 message.

Example

The following example finds all the entries in the validation list object WEBUSRS.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 52.

```
#include <qsyvldl.h>
#include <errno.h>

main()
{
  #define VLD_LST  "WEBUSRS   WEBLIB    "
  Qsy_Rtn_Vld_Lst_Ent_T  entry_1;
  Qsy_Rtn_Vld_Lst_Ent_T  entry_2;
  Qsy_Rtn_Vld_Lst_Ent_T  *input_info,
                         *output_info,
                         *temp;
  Qsy_Entry_ID_Info_T    *input_entry;
  short int              i;
  int                    rtn_errno;

  /* Initialize pointers to input and output buffers.              */
  output_info = addr(entry_1);
  input_info = addr(entry_2);
  /* Get the first entry in the validation list.                   */
  rtn_errno = QsyFindFirstValidationLstEntry(
                            (Qsy_Qual_Name_T *)&VLD_LST,
                             output_info))

  while (0 == rtn_errno)
  { /* Process all the entries in the validation list.             */
          .
          .
          .
     (process the entry)
          .
          .
          .
    /* Switch the pointers to the buffers so that the output from   */
    /* the last find operation is used as input to the 'find-next'
    /* operation.                                                   */
    temp = output_info;
    output_info = input_info;
    input_info = temp;

    /* Find the next entry.                                        */
    rtn_errno = QsyFindNextValidationLstEntry(
                             (Qsy_Qual_Name_T *)&VLD_LST,
                             &(input_info->Entry_ID_Info),
                             output_info))
  }
  /* Check if an error occurred.                                   */
  if (0 != rtn_errno && ENOREC != errno)
    perror("Find of validation list entry");

}
```

API introduced: V4R1

Top | Security APIs | APIs by category

Find Validation List Entry (QSYFDVLE) API

Required Parameter Group:

1	Qualified validation list name	Input	Char(20)
2	Entry ID information	Input	Char(*)

3	Attribute information	Input	Char(*)
4	Return entry	Output	Char(1724)
5	Return attributes	Output	Char(*)
6	Error Code	I/O	Char(*)

Service Program Name: QSYVLDL
Default Public Authority: *USE
Threadsafe: Yes

The Find Validation List Entry (QSYFDVLE) API finds an entry in a validation list object and returns it. Also, any attributes associated with the entry can be returned. To find an entry, there must be an exact match in the entry for the value that is specified in the entry ID parameter and the length of the entry ID. For example, an entry ID value of SMITH with a length of 5 would not find an entry where the entry ID is SMITH and the length is 7.

Authorities and Locks

Validation List Object
*USE

Validation List Object Library
*EXECUTE

Note: If the QsyEncryptData attribute is set to 1, then the user must have *USE, *ADD, and *UPD authorities to the validation list to get the data to be encrypted returned in the encrypted data field.

Required Parameter Group

Qualified validation list name
INPUT; CHAR(20)

The qualified object name of the validation list in which to find the entry. The first 10 characters specify the validation list name, and the second 10 characters specify the library.

You can use these special values for the library name:

*CURLIB	The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.
*LIBL	The library list is used to locate the validation list.

Entry ID information
INPUT; CHAR(*)

The format of the entry ID information is as follows. See "Field Descriptions" on page 30 for more information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of entry ID
4	4	BINARY(4)	CCSID of entry ID
8	8	CHAR(*)	Entry ID

Attribute information
INPUT; CHAR(*)

The format of the attribute information is as follows. See "Field Descriptions" on page 30 for more information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number of attributes
4	4	CHAR(*)	Attribute structures

The format of the attribute structure is as follows. See "Field Descriptions" on page 30 for more information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of attribute entry
4	4	BINARY(4)	Attribute location
8	8	BINARY(4)	Attribute type
12	C	BINARY(4)	Displacement to attribute ID
16	10	BINARY(4)	Length of attribute ID
20	14	BINARY(4)	Bytes provided for attribute
		CHAR(*)	Attribute ID

Return entry

OUTPUT; CHAR(1724)

The format of the return entry information is as follows. See "Field Descriptions" on page 30 for more information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of entry ID
4	4	BINARY(4)	CCSID of entry ID
8	8	CHAR(100)	Entry ID
108	6C	BINARY(4)	Length of encrypted data
112	70	BINARY(4)	CCSID of encrypted data
116	74	CHAR(600)	Encrypted data
716	2CC	BINARY(4)	Length of data
720	2D0	BINARY(4)	CCSID of data
724	2D4	CHAR(1000)	Data
1724	6BC	CHAR(20)	Reserved

Return attributes

OUTPUT; CHAR(*)

The format of the return attributes information is as follows. See "Field Descriptions" on page 30 for more information.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of attribute entry
4	4	BINARY(4)	Bytes returned
8	8	BINARY(4)	Bytes available
12	C	BINARY(4)	Length of attribute
16	10	BINARY(4)	CCSID of attribute
20	14	CHAR(*)	Attribute value

The size of this buffer must be 24 bytes multiplied by the number of attributes, plus the bytes provided in the buffer for each attribute. For example, if you are requesting 2 attributes and providing 8 bytes for one attribute and 5 bytes for the other attribute, you would need a 61-byte buffer. If the buffer is not large enough, the results are unpredictable.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

Field Descriptions

Attribute ID. The ID of the attribute. For system-defined attributes, the allowed values are:

String value	Description
QsyEncryptData	This is the attribute that is associated with the data to encrypt.
QsyEntryUsage	This is the entry usage information attribute.
QsyX509Cert	This is the X.509 certificate attribute for the entry.

Attribute location. Where the attribute is stored. The allowed value is:

0 The attribute is stored in the validation list object.

Attribute structures. Zero or more attribute structures that define the attributes that are associated with the entry.

Attribute type. The type of attribute. The allowed value follows:

0 This is a system-defined attribute.

Attribute value. The value of the returned attribute. If the attribute ID is QsyEncryptData or QsyX509Cert, the data will be in the form of variable length character array. If the attribute ID is QsyEntryUsage, the data will be in the form of Qsy_Rtn_Entry_Usage_Attr_T.

The format of the Qsy_Rtn_Entry_Usage_Attr_T structure is as follows. See “Field Descriptions” for more information.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(8)	Create date

Offset		Type	Field
Dec	Hex		
8	8	CHAR(8)	Last used date
16	10	CHAR(8)	Encrypted data change date
24	18	BINARY(4)	Not valid verify count

Bytes available. The number of bytes of data that is available to be returned to the user for the current attribute. If all data is returned, bytes available is the same as the number of bytes returned. If the bytes available is 16, then the specified attribute is not defined for this entry.

Bytes provided for attribute. The number of bytes provided in the return attributes buffer for the attribute value. The minimum length is 0. If 0 is specified, the bytes available will indicate if the attribute exists and how many bytes of data are needed to return the attribute.

Bytes returned. The number of bytes of data that is returned to the user for the current attribute. This is the lesser of the number of bytes available to be returned and bytes provided for attribute plus 20.

CCSID of attribute. An integer that represents the CCSID for the attribute. Valid CCSID values are in the range 0 through 65535. This value is the CCSID value that was specified when the attribute was added or changed. If the value is 0, then no CCSID value was stored with the attribute.

CCSID of encrypted data. An integer that represents the CCSID for the encrypted data.

CCSID of data. An integer that represents the CCSID for the data.

CCSID of entry ID. An integer that represents the CCSID for the entry ID. Valid CCSID values are in the range 0 through 65535. This field is not used to find the entry. The value is returned in the return entry.

Create date. The date the entry was added to the validation list, in *DTS (date-time stamp) format.

Data. The data that is stored in the validation list entry.

Displacement to attribute ID. The displacement in the attribute entry to the start of the attribute ID.

Encrypted data. If the QsyEncryptData attribute for this entry is 1 and the QRETSVRSEC system value is '1', then the encrypted data that is stored in the entry will be decrypted and returned in this field. If the QsyEncryptData attribute is 0 or the QRETSVRSEC system value is '0', then the encrypted data cannot be returned and the contents of this field are unpredictable.

Encrypted data change date. The date the encrypted data was last changed, in *DTS (date-time stamp) format.

Entry ID. The data that is used to find the entry in the validation list.

Last used date. The date of the last successful verify, in *DTS (date-time stamp) format.

Length of attribute. The length (in bytes) of the returned attribute value. This value will be less than or equal to the bytes provided for attribute.

Length of attribute entry. The length (in bytes) of the current entry. This length can be used to access the next entry, and must be a multiple of 4.

Length of attribute ID. The number of bytes of data in the attribute ID. The length must be greater than 0.

Length of data. The number of bytes of data that is stored in this validation list entry. Possible values are 0 to 1000.

Length of encrypted data. The number of bytes of encrypted data that is stored in this validation list entry. Possible values are 0 to 600. If the QsyEncryptData attribute is 0 or the QRETSVRSEC system value is '0', then the length will always be 0.

Length of entry ID. The number of bytes of data that is provided as the entry ID. Possible values are 1 through 100.

Not valid verify count. The number of times that incorrect encrypted data has been specified on a verify since the last successful verify.

Number of attributes. The number of attributes to be returned. This value must be greater than or equal to 0. If the value is 0, then no attributes will be returned.

Reserved. This is an ignored field.

Error Messages

Message ID	Error Message Text
CPFA0AA E	Error occurred while attempting to obtain space.
CPF226B E	Validation list entry does not exist.
CPF3CF1 E	Error code parameter not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9804 E	Object &2 in library &3 damaged.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R2

Top | Security APIs | APIs by category

QsyFindValidationLstEntry()—Find Validation List Entry API

Syntax

```
#include <qsyvldl.h>

int QsyFindValidationLstEntry
      (Qsy_Qual_Name_T        *Validation_Lst,
       Qsy_Entry_ID_Info_T    *Entry_ID,
       Qsy_Rtn_Vld_Lst_Ent_T  *Rtn_Entry);
```

Service Program Name: QSYVLDL
Default Public Authority: *USE
Threadsafe: Yes

The **QsyFindValidationLstEntry()** function finds an entry in a validation list object. The function then returns the information for the entry in the buffer that is pointed to by the *Rtn_Entry* parameter. To find an entry, there must be an exact match in the entry for the value that is specified in the *Entry_ID*

parameter and the length of the entry ID. For example, an entry ID value of ″SMITH″ with a length of 5 would not find an entry where the entry ID is ″SMITH ″ and the length is 7.

Authorities

Validation List Object
*USE

Validation List Object Library
*EXECUTE

Note: If the QsyEncryptData attribute is set to QSY_VFY_FIND_E (1), then the user must have *USE, *ADD, and *UPD authority to the validation list to get the data to be encrypted returned in the *Rtn_Entry* parameter.

Parameters

Validation_Lst
(Input)

A pointer to the qualified object name of the validation list in which to find the entry. The first 10 characters specify the validation list name, and the second 10 characters specify the library. You can use these special values for the library name:

**CURLIB*	The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.
**LIBL*	The library list is used to locate the validation list.

Entry_ID
(Input)

A pointer to the entry ID information. The format of the Qsy_Entry_ID_Info_T structure is as follows:

int	Entry_ID_Len	The number of bytes of data that is provided as the entry ID. Possible values are from 1 through 100.
unsigned int	Entry_ID_CCSID	An integer that represents the CCSID for the entry ID. Valid CCSID values are in the range 0 through 65535. This value is not used to find the entry.
unsigned char	Entry_ID[100]	The data that is used to identify this entry in the validation list.

Rtn_Entry
(Output)

A pointer to the buffer where the entry information is placed. The buffer must be allocated to the size of the *Qsy_Rtn_Vld_Lst_Ent_T* structure or the results will be unpredictable. The format of the *Qsy_Rtn_Vld_Lst_Ent_T* structure is as follows:

Qsy_Entry_ID_Info_T	Entry_ID_Info	The entry ID information structure.
Qsy_Entry_Encr_Data_Info_T	Encr_Data_Info	The data to be encrypted information structure.
Qsy_Entry_Data_Info_T	Entry_Data_Info	The entry data information structure.
char	Reserved[4]	This is an ignored field.
void *	Entry_More_Info	A pointer to additional information. This pointer is currently set to NULL.

See the Entry_ID (page 33) parameter for the format of the Qsy_Entry_ID_Info_T structure.

The format of the Qsy_Entry_Encr_Data_Info_T structure is as follows:

int	Encr_Data_Len	The number of bytes of encrypted data that is stored in this validation list entry. If the QsyEncryptData attribute is 0 or the QRETSVRSEC system value is '0', the length will always be 0.
unsigned int	Encr_Data_CCSID	The CCSID associated with the encrypted data.
unsigned char	Encr_Data[600]	If the QsyEncryptData attribute is 1 and the QRETSVRSEC system value is '1', then the encrypted data that is stored in the entry will be decrypted and returned in this field. If the QsyEncryptData attribute is 0 or the QRETSVRSEC system value is '0', then the encrypted data cannot be returned, and the contents of this field are unpredictable.

The format of the Qsy_Entry_Data_Info_T structure is as follows:

int	Entry_Data_Len	The length of the entry data.
unsigned int	Entry_Data_CCSID	The CCSID associated with the entry data.
unsigned char	Entry_Data[1000]	The entry data.

Return Value

0 **QsyFindValidationLstEntry()** was successful. The return value points to the entry.
-1 **QsyFindValidationLstEntry()** was not successful. The *errno* global variable is set to indicate the error.

Error Conditions

If **QsyFindValidationLstEntry()** is not successful, *errno* indicates one of the following errors:

3401 [EACCES]

The current user does not have *USE authority to the validation list object, or does not have *EXECUTE authority to the validation list object library.

3406 [EAGAIN]

The validation list object is currently locked by another process.

3484 [EDAMAGE]

The validation list object is damaged.

3021 [EINVAL]

Parameter value is not valid.

3025 [ENOENT]

The validation list object was not found.

3026 [ENOREC]

Specified entry does not exist.

3474 [EUNKNOWN]

Unknown system state. Check the job log for a CPF9872 message.

Example

The following example finds all the entries in the validation list object WEBUSRS where the entry ID starts with 'abc'.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 52.

```
#include <qsyvldl.h>
#include <errno.h>

main()
{
  #define VLD_LST  "WEBUSRS   WEBLIB    "
  Qsy_Rtn_Vld_Lst_Ent_T  entry_1;
  Qsy_Rtn_Vld_Lst_Ent_T  entry_2;
  Qsy_Rtn_Vld_Lst_Ent_T  *input_info,
                         *output_info,
                         *temp;
  Qsy_Entry_ID_Info_T    *input_entry;
  short int              i;
  int                    rtn_errno;

  /* Set up entry ID to find.                                        */
  strncpy(entry_1.Entry_ID_Info.Entry_ID,"abc",3);
  entry_1.Entry_ID_Info.Entry_ID_Len = 3;

  /* Initialize pointers to input and output buffers.                */
  input_info = addr(entry_1);
  output_info = addr(entry_2);

  /* Try to find an entry for 'abc'.                                 */
  rtn_errno = QsyFindValidationLstEntry(
                       (Qsy_Qual_Name_T *)&VLD_LST,
                       &entry_1.Entry_ID_Info,
                       output_info))
  /* If an 'abc' entry does not exist.                               */
  if (0 != rtn_errno && ENOREC == errno)
    /* Find the next entry after 'abc'.                              */
    rtn_errno = QsyFindNextValidationLstEntry(
                              (Qsy_Qual_Name_T *)&VLD_LST,
                              &entry_1.Entry_ID_Info,
                              output_info))

  while (0 == rtn_errno &&
         3 <= output_info->Entry_ID_Info.Entry_ID_Len &&
         0 == strncmp(output_info->Entry_ID_Info.Entry_ID,"abc",3))
  { /* Process all the entries in the validation list that           */
    /* begin with 'abc'.                                             */
          .
          .
          .
     (process the entry)
          .
          .
          .
    /* Switch the pointers to the buffers so that the output from    */
    /* the last find operation is used as input to the 'find-next'   */
    /* operation.                                                    */
    temp = output_info;
    output_info = input_info;
    input_info = temp;

    /* Find the next entry.                                          */
    rtn_errno = QsyFindNextValidationLstEntry(
                              (Qsy_Qual_Name_T *)&VLD_LST,
                              &(input_info->Entry_ID_Info),
```

```
                            output_info))
  }
  /* Check if an error occurred.                                    */
  if (0 != rtn_errno && ENOREC != errno)
    perror("Find of validation list entry");

}
```

API introduced: V4R1

Top | Security APIs | APIs by category

QsyFindValidationLstEntryAttrs()—Find Validation List Entry Attributes API

Syntax

```
#include <qsyvldl.h>

int QsyFindValidationLstEntryAttrs
       (Qsy_Qual_Name_T           *Validation_Lst,
        Qsy_Entry_ID_Info_T       *Entry_ID,
        Qsy_Rtn_Vld_Lst_Ent_T     *Rtn_Entry,
        Qsy_Attr_Info_T           *Rtn_Attributes);
```

Service Program Name: QSYVLDL
Default Public Authority: *USE
Threadsafe: Yes

The **QsyFindValidationLstEntryAttrs()** function finds an entry in a validation list object, and the attributes associated with the entry. The function then returns the information for the entry in the buffer that is pointed to by the *Rtn_Entry* parameter, and the information for the attributes in the buffer that is pointed to by the *Rtn_Attributes* parameter. To find an entry, there must be an exact match in the entry for the value that is specified in the *Entry_ID* parameter and the length of the entry ID. For example, an entry ID value of ″SMITH″ with a length of 5 would not find an entry where the entry ID is ″SMITH ″ and the length is 7.

Authorities

Validation List Object
*USE

Validation List Object Library
*EXECUTE

Note: If the QsyEncryptData attribute is set to QSY_VFY_FIND (1), then the user must have *USE, *ADD, and *UPD authority to the validation list to get the data to be encrypted returned in the *Rtn_Entry* parameter.

Parameters

Validation_Lst
(Input)

A pointer to the qualified object name of the validation list in which to find the entry. The first 10 characters specify the validation list name, and the second 10 characters specify the library. You can use these special values for the library name:

**CURLIB* The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.

**LIBL* The library list is used to locate the validation list.

Entry_ID

(Input)

A pointer to the entry ID information. The format of the Qsy_Entry_ID_Info_T structure is as follows:

int	Entry_ID_Len	The number of bytes of data that is provided as the entry ID. Possible values are from 1 through 100.
unsigned int	Entry_ID_CCSID	An integer that represents the CCSID for the entry ID. Valid CCSID values are in the range 1 through 65535. This value is not used to find the entry.
unsigned char	Entry_ID[]	The data that is used to identify this entry in the validation list.

Rtn_Entry

(Output)

A pointer to the buffer where the entry information is placed. The buffer must be allocated to the size of the *Qsy_Rtn_Vld_Lst_Ent_T* structure or the results will be unpredictable. The format of the *Qsy_Rtn_Vld_Lst_Ent_T* structure is as follows:

Qsy_Entry_ID_Info_T	Entry_ID_Info	The entry ID information structure.
Qsy_Entry_Encr_Data_Info_T	Encr_Data_Info	The data to be encrypted information structure.
Qsy_Entry_Data_Info_T	Entry_Data_Info	The entry data information structure
char	Reserved[4]	This is an ignored field.
void *	Entry_More_Info	A pointer to additional information. This pointer is currently set to NULL.

See the Entry_ID (page 37) parameter for the format of the Qsy_Entry_ID_Info_T structure.

The format of the Qsy_Entry_Encr_Data_Info_T structure is as follows:

int	Encr_Data_Len	The number of bytes of encrypted data that is stored in this validation list entry. If the QsyEncryptData attribute is 0 or the QRETSVRSEC system value is '0', the length will always be 0.
unsigned int	Encr_Data_CCSID	The CCSID associated with the encrypted data.
unsigned char	Encr_Data[600]	If the QsyEncryptData attribute is 1 and the QRETSVRSEC system value is '1', then the encrypted data that is stored in the entry will be decrypted and returned in this field. If the QsyEncryptData attribute is 0 or the QRETSVRSEC system value is '0', then the encrypted data cannot be returned, and the contents of this field are unpredictable.

The format of the Qsy_Entry_Data_Info_T structure is as follows:

int	Entry_Data_Len	The length of the entry data.
unsigned int	Entry_Data_CCSID	The CCSID associated with the entry data.

unsigned char	Entry_Data[1000]	The entry data.

Rtn_Attributes

(Input) A pointer to a structure that indicates the attributes to return. The format of the Qsy_Attr_Info_T structure is as follows:

int	Number_Attrs	The number of attributes to be returned. This value must be greater than 0.
char	Res_Align[12]	Reserved for boundary alignment.
Qsy_Attr_Descr_T	Attr_Descr[]	An array of attribute description structures.

The format of the Qsy_Attr_Descr_T structure is as follows:

int	Attr_Location	Where the attribute is stored. The allowed value follows: *0 QSY_IN_VLDL* The attribute is stored in the validation list object.
int	Attr_Type	The type of attribute. The allowed value follows: *0 QSY_SYSTEM_ATTR* This is a system-defined attribute.
union	Attr_Res Res_1[8]	Reserved data. This value must be hexadecimal zero.
char *	Attr_ID	The ID of the attribute. For system-defined attributes, the allowed values are:
	String value	Description
	QsyEncryptData	This is the attribute that is associated with the data to encrypt.
	QsyX509Cert	This is the X.509 certificate attribute for the entry.
	QsyEntryUsage	This is the entry usage information attribute.
union	Attr_Other_Descr Res_1[32]	Reserved data. This value must be hexadecimal zero.
union	Attr_Data_Info	The information that describes the attribute data.
union	Attr_Other_Data Res_1[32]	Reserved data. This value must be hexadecimal zero.

The format of the Attr_Data_Info union is as follows:

Qsy_In_VLDL_T	Attr_VLDL	The attribute data information for an attribute that is stored in the validation list object.
union	Attr_In_Other Res_1[96]	Reserved data. The last 64 bytes must be hexadecimal zero.

The format of the Qsy_In_VLDL_T structure is as follows:

int	Attr_CCSID	An integer that represents the CCSID for the attribute. Valid CCSID values are in the range -1 through 65535. This value is not used.
int	Attr_Len	The number of bytes of data in the buffer to return the attribute value. The minimum length is 12.
union	Attr_Res Res_1[8]	Reserved data. This value must be hexadecimal zero.
void *	Attr_Value	A pointer to a Qsy_Rtn_VLDL_Attr_T structure in which to return the attribute.

The format of the Qsy_Rtn_VLDL_Attr_T structure is as follows:

int	Bytes_Returned	The number of bytes of data that is returned to the user in the attribute buffer. This is the lesser of the number of bytes available to be returned and Attr_Len in Qsy_In_VLDL_T.
int	Bytes_Available	The number of bytes of data that is available to be returned to the user in the attribute buffer. If all data is returned, bytes available is the same as the number of bytes returned. If the bytes available is 12, then the specified attribute is not defined for this entry.
int	Attr_Len	The length (in bytes) of the returned attribute.
unsigned int	Attr_CCSID	An integer that represents the CCSID for the attribute. Valid CCSID values are in the range 0 through 65535. This value is the CCSID value that was specified when the attribute was added or changed. If the value is 0, then no CCSID value was stored with the attribute.
unsigned char	Attr_Data[]	The value of the returned attribute.

The format of the Qsy_Rtn_Entry_Usage_Attr_T structure is as follows:

char	Create_Date[8]	The date the entry was added to the validation list.
char	Last_Used_Date[8]	The date of the last successful verify.
char	Encr_Data_Chg_Date[8]	The date the encrypted data was last changed.
int	Not_Valid_Verify_Count	The number of times that incorrect encrypted data has been specified on a verify since the last successful verify.

Return Value

0 **QsyFindValidationLstEntryAttrs()** was successful. The return value points to the entry. The return attribute points to the attribute list.

-1 **QsyFindValidationLstEntryAttrs()** was not successful. The *errno* global variable is set to indicate the error.

Error Conditions

If **QsyFindValidationLstEntryAttrs()** is not successful, *errno* indicates one of the following errors:

3401 [EACCES]

The current user does not have *USE authority to the validation list object, or does not have *EXECUTE authority to the validation list object library.

3406 [EAGAIN]

The validation list object is currently locked by another process.

3484 [EDAMAGE]

The validation list object is damaged.

3021 [EINVAL]

Parameter value is not valid.

3025 [ENOENT]

The validation list object was not found.

3026 [ENOREC]

Specified entry does not exist.

3404 [ENOSPC]

No space available.

3474 [EUNKNOWN]

Unknown system state. Check the job log for a CPF9872 message.

Example

The following example finds an entry for a user named FRED in the validation list object WEBUSRS, and returns the attribute that is associated with the encrypted data field.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 52.

```
#include <stdlib.h>
#include <qsyvldl.h>
#include <errno.h>

main()
{
  #define VLD_LST  "WEBUSRS   WEBLIB    "
  Qsy_Rtn_Vld_Lst_Ent_T   rtn_ent;
  struct {
    Qsy_Attr_Info_T       attr_info;
    Qsy_Attr_Desc_T       attr_desc;
  } rtn_attr;
  struct {
    Qsy_Rtn_VLDL_Attr_T   encr_info;
    char                  encr_val;
  } encr_attr;
  Qsy_Entry_ID_Info_T    *input_entry;

  /* Set up entry ID to find.                                    */
  strncpy(rtn_ent.Entry_ID_Info.Entry_ID,"FRED",4);
  rtn_ent.Entry_ID_Info.Entry_ID_Len = 4;

  /* Set up the attribute information.                           */
                                /* Initialize reserved fields.    */
  memset(rtn_attr.attr_desc.Attr_Res.Res_1,
         0,
         sizeof(rtn_attr.attr_desc.Attr_Res.Res_1));
  memset(rtn_attr.attr_desc.Attr_Other_Descr.Res_1,
         0,
         sizeof(rtn_attr.attr_desc.Attr_Other_Descr.Res_1));
  memset(rtn_attr.attr_desc.Attr_Data_Info.Attr_In_Other.Res_1,
         0,
         sizeof(rtn_attr.attr_desc.Attr_Data_Info.Attr_In_Other.Res_1));
```

```
  memset(rtn_attr.attr_desc.Attr_Other_Data.Res_1,
         0,
         sizeof(rtn_attr.attr_desc.Attr_Other_Data.Res_1));

                                  /* Set number of attrs.           */
  rtn_attr.attr_info.Numbers_Attrs = 1;
                                  /* Set location of attribute.     */
  rtn_attr.attr_desc.Attr_Location = QSY_IN_VLDL;
                                  /* Set attribute type.            */
  rtn_attr.attr_desc.Attr_Type = QSY_SYSTEM_ATTR;
                                  /* Set attribute type.            */
  rtn_attr.attr_desc.Attr_ID = (char *)QSY_ENCRYPT_DATA;
                                  /* Set length to retrieve.        */
  rtn_attr.attr_desc.Attr_Data_Info.Attr_VLDL.Attr_Len =
                                 sizeof(encr_attr);
                                  /* Set CCSID value.               */
  rtn_attr.attr_desc.Attr_Data_Info.Attr_VLDL.Attr_CCSID = -1;
                                  /* Set pointer to return buffer   */
  rtn_attr.attr_desc.Attr_Data_Info.Attr_VLDL.Attr_Value =
    (void *)&encr_attr;

  /* Try to find an entry for 'FRED'.                               */
  if (0 == QsyFindValidationLstEntryAttrs(
                     (Qsy_Qual_Name_T *)&VLD_LST,
                     &rtn_ent.Entry_ID_Info,
                     &rtn_ent,
                     (Qsy_Attr_Info_T *)&rtn_attr))
  { /* Entry was found                                              */
          .
          .
          .
     (process the entry)
          .
          .
          .
  }
  else /* Error on find of entry.                                   */
    perror("Find of validation list entry");

}
```

API introduced: V4R2

Top | Security APIs | APIs by category

Open List of Validation List Entries (QSYOLVLE) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	List information	Output	Char(80)
4	Number of records to return	Input	Binary(4)
5	Format name	Input	Char(8)
6	Qualified validation list name	Input	Char(20)
7	Error code	I/O	Char(*)

Default Public Authority: *USE
Threadsafe: No

The Open List of Validation List Entries (QSYOLVLE) API returns a list of validation list entries in a validation list object. Upon successful completion of this API, a handle is returned in the list information parameter. You may use this handle on subsequent calls to the following APIs:

- Get List Entries (QGYGTLE)
- Find List Entry (QGYFNDE)
- Close List (QGYCLST)

Authorities and Locks

Authority to Validation List
*USE

Authority to Validation List Library
*EXECUTE

Required Parameter Group

Receiver variable
OUTPUT; CHAR(*)

The receiver variable that receives the information requested. You can specify the size of the area to be smaller than the format requested as long as you specify the length parameter correctly. As a result, the API returns only the data that the area can hold.

Length of receiver variable
INPUT; BINARY(4)

The length of the receiver variable. If the length is larger than the size of the receiver variable, the results are not predictable.

List Information
OUTPUT; CHAR(80)

Information about the list that is created by this program. See "Format of List information" on page 43 for a description of the layout of this parameter.

Number of records to return
INPUT; BINARY(4)

The number of records in the list to put into the receiver variable. Possible values follow:

-1	The entire list is built synchronously.
0	The entire list is built asynchronously in a server job.
Positive number of records	At least that many records will be built synchronously and the remainder will be built asynchronously in a server job.

Format name
INPUT; CHAR(8)

The name of the format that is used to return information about the validation list entries.

You can specify these formats:

"VLDE0100 Format" on page 44	The order and format of the data that is returned in the receiver variable for each validation list entry in the list.

Qualified validation list name
INPUT; CHAR(20)

The qualified object name of the validation list that contains the entries to return. The first 10 characters specify the validation list name, and the second 10 characters specify the library. You

can use these special values for the library name:

**CURLIB* The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.
**LIBL* The library list is used to locate the validation list.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

Format of List information

For detailed descriptions of the fields in the tables, see "Field Descriptions."

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Total records
4	4	BINARY(4)	Records returned
8	8	CHAR(4)	Request handle
12	C	BINARY(4)	Record length
16	10	CHAR(1)	Information complete indicator
17	11	CHAR(13)	Date and time created
30	1E	CHAR(1)	List status indicator
31	1F	CHAR(1)	Reserved
32	20	BINARY(4)	Length of information returned
36	24	BINARY(4)	First record in buffer
40	28	CHAR(40)	Reserved

Field Descriptions

Date and time created. The date and time when the list was created. The 13 characters are:

1 Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
2-7 The date, in YYMMDD (year, month, and day) format.
8-13 The time of day, in HHMMSS (hours, minutes, and seconds) format.

First record in buffer. The number of the first record in the receiver variable.

Information complete indicator. Whether all information that was requested has been supplied.

I Incomplete information. An interruption causes the list to contain incomplete information about a buffer or buffers.
P Partial and accurate information. Partial information is returned when the maximum space was used and not all of the buffers requested were read.
C Complete and accurate information. All the buffers requested are read and returned.

Length of information returned. The size in bytes of the information returned in the receiver variable.

List status indicator. The status of building the list. Possible values follow:

0 The list building is pending.
1 The list is in the process of being built.
2 The list has been completely built.
3 An error occurred when building the list. An error will be signalled to the caller of the QGYGTLE API.
4 The list is primed and ready to be built.

Record length. The length of each record of information returned. This value will be set to 0 because the record lengths are variable. You can obtain the length of individual records from the records themselves.

Records returned. The number of records returned in the receiver variable.

This is the smallest of the following three values:

- The number of records that fit into the receiver variable.
- The number of records in the list.
- The number of records that are requested.

Request handle. The handle of the request that can be used for subsequent requests of information from the list. The handle is valid until the Close List (QGYCLST) API is called to close the list, or until the job ends.

Note: This field should be treated as a hexadecimal field. It should not be converted from one CCSID to another, for example, EBCDIC to ASCII, because doing so could result in an unusable value.

Reserved. An ignored field.

Total records. The total number of records available in the list.

VLDE0100 Format

The following table describes the order and format of the data that is returned in the receiver variable for each validation list entry in the list. For detailed descriptions of the fields in the table, see "Field Descriptions" on page 45.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of entry
4	4	BINARY(4)	Displacement to entry ID
8	8	BINARY(4)	Length of entry ID
12	C	BINARY(4)	CCSID of entry ID
16	10	BINARY(4)	Displacement to encrypted data
20	14	BINARY(4)	Length of encrypted data
24	18	BINARY(4)	CCSID of encrypted data
28	1C	BINARY(4)	Displacement to entry data
32	20	BINARY(4)	Length of entry data
36	24	BINARY(4)	CCSID of entry data
		CHAR(*)	Entry ID
		CHAR(*)	Encrypted data
		CHAR(*)	Entry data

Field Descriptions

CCSID of encrypted data. The CCSID of the encrypted data that was specified when the validation list entry was added or changed.

CCSID of entry data. The CCSID of the entry data that was specified when the validation list entry was added or changed.

CCSID of entry ID. The CCSID of the entry ID that was specified when the validation list entry was added.

Displacement to encrypted data. The displacement in the entry to the start of the encrypted data.

Displacement to entry data. The displacement in the entry to the start of the entry data.

Displacement to entry ID. The displacement in the entry to the start of the entry ID.

Encrypted data. The encrypted data associated with the validation list entry. This data is only returned if the entry specifies that the encrypted data is two way encrypted, the QRETSVRSEC system value is '1', and the user has *USE, *ADD, and *UPD authority to the validation list. If the data is to be returned, it is decrypted and returned in this field.

Entry data. The data associated with the validation list entry.

Entry ID. The entry ID for the validation list entry.

Length of encrypted data. The length (in bytes) of the encrypted data. If the data is one-way encrypted, the QRETSVRSEC system value is '0', or the user is not authorized to have the encrypted data returned, this value will be 0.

Length of entry. The length (in bytes) of the current entry. This length can be used to access the next entry.

Length of entry data. The length (in bytes) of the entry data.

Length of entry ID. The length (in bytes) of the entry ID.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF226B E	Validation list entry does not exist.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9804 E	Object &2 in library &3 damaged.
CPF9821 E	Not authorized to program &1 in library &2.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
GUI0002 E	&2 is not valid for length of receiver variable.
GUI0027 E	&1 is not valid for number of records to return.

API introduced: V4R2

QsyRemoveValidationLstEntry()—Remove Validation List Entry API

Syntax

```
#include <qsyvldl.h>

int QsyRemoveValidationLstEntry
       (Qsy_Qual_Name_T        *Validation_Lst,
        Qsy_Entry_ID_Info_T    *Entry_ID);
```

Service Program Name: QSYVLDL
Default Public Authority: *USE
Threadsafe: Yes

The **QsyRemoveValidationLstEntry()** function removes an entry from a validation list object. To identify an entry to be removed, there must be an exact match in the entry for the value that is specified in the *Entry_ID* parameter and the length of the entry ID. For example, an entry ID value of ″SMITH″ with a length of 5 would not remove an entry where the entry ID is ″SMITH ″ and the length is 7.

Authorities

Validation List Object
*USE and *DLT

Validation List Object Library
*EXECUTE

Parameters

Validation_Lst
(Input)

A pointer to the qualified object name of the validation list that contains the entry to remove. The first 10 characters specify the validation list name, and the second 10 characters specify the library. You can use these special values for the library name:

**CURLIB*	The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.
**LIBL*	The library list is used to locate the validation list.

Entry_ID
(Input)

A pointer to the entry ID information. Qsy_Entry_ID_Info_T structure is as follows:

int	Entry_ID_Len	The number of bytes of data that is provided as the entry ID. Possible values are from 1 through 100.
unsigned int	Entry_ID_CCSID	An integer that represents the CCSID for the entry ID. Valid CCSID values are in the range 0 through 65535. This value is not used to remove the entry.
unsigned char	Entry_ID[100]	The data that is used to identify this entry in the validation list.

Return Value

0 **QsyRemoveValidationLstEntry()** was successful.

-1 QsyRemoveValidationLstEntry() was not successful.

The *errno* global variable is set to indicate the error.

Error Conditions

If **QsyRemoveValidationLstEntry()** is not successful, *errno* indicates one of the following errors:

3401 [EACCES]

The current user does not have *USE and *DLT authorities to the validation list object, or does not have *EXECUTE authority to the validation list object library.

3406 [EAGAIN]

The validation list object is currently locked by another process.

3484 [EDAMAGE]

The validation list object is damaged.

3021 [EINVAL]

Parameter value is not valid.

3025 [ENOENT]

The validation list object was not found.

3026 [ENOREC]

Specified entry does not exist.

3474 [EUNKNOWN]

Unknown system state. Check the job log for a CPF9872 message.

Example

The following example removes an entry for a user named FRED in the validation list object WEBUSRS.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 52.

```
#include <qsyvldl.h>

main()
{
  #define VLD_LST  "WEBUSRS   WEBLIB    "
  Qsy_Entry_ID_Info_T    entry_info;

  entry_info.Entry_ID_Len = 4;
  strncpy(entry_info.Entry_ID,"FRED",entry_info.Entry_ID_Len);

  if (0 != QsyRemoveValidationLstEntry(
                       (Qsy_Qual_Name_T *)&VLD_LST,
                       &entry_info))
     perror("QsyRemoveValidationLstEntry()");

}
```

API introduced: V4R1

Top | Security APIs | APIs by category

Remove Validation List Entry (QSYRMVLE) API

Required Parameter Group:

1	Qualified validation list name	Input	Char(20)
2	Entry ID information	Input	Char(*)
3	Error code	I/O	Char(*)

Service Program Name: QSYVLDL
Default Public Authority: *USE
Threadsafe: Yes

The Remove Validation List Entry (QSYRMVLE) API removes an entry from a validation list object. To identify an entry to be removed, there must be an exact match in the entry for the value that is specified in the entry ID parameter and the length of the entry ID. For example, an entry ID value of ″SMITH″ with a length of 5 would not remove an entry where the entry ID is ″SMITH ″ and the length is 7.

Authorities and Locks

Validation List Object
*USE and *DLT

Validation List Object Library
*EXECUTE

Required Parameter Group

Qualified validation list name
INPUT; CHAR(20)

The qualified object name of the validation list that contains the entry to remove. The first 10 characters specify the validation list name, and the second 10 characters specify the library. You can use these special values for the library name:

**CURLIB*	The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.
**LIBL*	The library list is used to locate the validation list.

Entry ID information
INPUT; CHAR(*)

The format of the entry ID information is as follows. See the “Field Descriptions” on page 49 for more information.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0	BINARY(4)	Length of entry ID
4	4	BINARY(4)	CCSID of entry ID
8	8	CHAR(*)	Entry ID

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.

Field Descriptions

CCSID of entry ID. An integer that represents the CCSID for the entry ID. Valid CCSID values are in the range 0 through 65535. This field is not used to remove the entry.

Entry ID. The data that is used to identify the entry to be removed from the validation list.

Length of entry ID. The number of bytes of data that is provided as the entry ID. Possible values are 1 through 100.

Error Messages

Message ID	Error Message Text
CPF226B E	Validation list entry does not exist.
CPF3CF1 E	Error code parameter not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9804 E	Object &2 in library &3 damaged.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V4R2

Top | Security APIs | APIs by category

QsyVerifyValidationLstEntry()—Verify Validation List Entry API

Syntax

```
#include <qsyvldl.h>

int QsyVerifyValidationLstEntry
       (Qsy_Qual_Name_T             *Validation_Lst,
        Qsy_Entry_ID_Info_T         *Entry_ID,
        Qsy_Entry_Encr_Data_Info_T  *Encrypt_Data);
```

Service Program Name: Name QSYVLDL
Default Public Authority: *USE
Threadsafe: Yes

The **QsyVerifyValidationLstEntry()** function verifies an entry in a validation list object. It verifies the entry by finding the validation list object, then finding the entry that is specified in the *Entry_ID* parameter. To find an entry, there must be an exact match in the entry for the value that is specified in the *Entry_ID* parameter and the length of the entry ID. For example, an entry ID value of ″SMITH″ with a length of 5 would not find an entry where the entry ID is ″SMITH ″ and the length is 7.

If the entry is found, the data specified in the *Encrypt_Data* parameter is encrypted by the system and compared to the encrypted data that is stored for the entry. If the encrypted data fields do not match, then -2 is returned by the function.

The verification of an entry should be done within the same process as the work that is being done on behalf of this entry ID so that there is accountability for the actions that are taken. Also, an entry ID should be verified just before the work is done on behalf of that entry ID, instead of verifying a set of entry IDs and then doing work on behalf of the different entry IDs.

Authorities

Validation List Object
*USE

Validation List Object Library
*EXECUTE

Parameters

Validation_Lst
(Input) A pointer to the qualified object name of the validation list that contains the entry to verify. The first 10 characters specify the validation list name, and the second 10 characters specify the library. You can use these special values for the library name:

**CURLIB*	The current library is used to locate the validation list. If there is no current library, QGPL (general purpose library) is used.
**LIBL*	The library list is used to locate the validation list.

Entry_ID
(Input)

A pointer to the entry ID information. The format of the Qsy_Entry_ID_Info_T structure is as follows:

int	Entry_ID_Len	The number of bytes of data that is provided as the entry ID. Possible values are from 1 through 100.
unsigned int	Entry_ID_CCSID	An integer that represents the CCSID for the entry ID. Valid CCSID values are in the range 1 through 65535. This field is not used to verify the entry.
unsigned char	Entry_ID[100]	The data that is used to identify this entry in the validation list.

Encrypt_Data
(Input)

A pointer to the encrypted data information that is associated with the entry ID. The format of the Qsy_Entry_Encr_Data_Info_T structure is as follows:

int	Encr_Data_Len	The number of bytes of data to be encrypted and compared to the encrypted data in the validation list entry. Possible values are 1 through 600.
unsigned int	Encr_Data_CCSID	An integer that represents the CCSID for the data to encrypt. Valid CCSID values are in the range 0 through 65535. This value is not used to verify the entry.
unsigned char	Encr_Data[600]	The data to be encrypted and compared to the encrypted data that is found for the specified entry ID in the validation list.

Return Value

0 QsyVerifyValidationLstEntry() was successful.
-1 QsyVerifyValidationLstEntry() was not successful.

The *errno* global variable is set to indicate the error.
-2 QsyVerifyValidationLstEntry() was not successful because the encrypted data was incorrect.

Error Conditions

If **QsyVerifyValidationLstEntry()** is not successful, *errno* indicates one of the following errors:

3401 [EACCES]

The current user does not have *USE authority to the validation list object, or does not have *EXECUTE authority to the validation list object library.

3406 [EAGAIN]

The validation list object is currently locked by another process.

3484 [EDAMAGE]

The validation list object is damaged.

3021 [EINVAL]

Parameter value is not valid.

3025 [ENOENT]

The validation list object was not found.

3026 [ENOREC]

Specified entry does not exist.

3474 [EUNKNOWN]

Unknown system state. Check the job log for a CPF9872 message.

Example

The following example validates the entry for a user named FRED in the validation list object WEBUSRS.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 52.

```
#include <qsyvldl.h>

main()
{
  #define VLD_LST  "WEBUSRS   WEBLIB    "
  Qsy_Entry_ID_Info_T    entry_info;
  Qsy_Entry_Encr_Data_Info_T  encrypt_data;

  entry_info.Entry_ID_Len = 4;
  strncpy(entry_info.Entry_ID,"FRED",entry_info.Entry_ID_Len);
  encrypt_data.Encr_Data_Len = 7;
  strncpy(encrypt_data.Encr_Data,"MSN1TJG",
            encrypt_data.Encr_Data_Len);

  if (0 != QsyVerifyValidationLstEntry((Qsy_Qual_Name_T *)&VLD_LST,
                                       &entry_info,
                                       &encrypt_data))
     perror("QsyVerifyValidationLstEntry()");

}
```

API introduced: V4R1

Top | Security APIs | APIs by category

Code license and disclaimer information

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM, ITS PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. LOSS OF, OR DAMAGE TO, DATA;
2. DIRECT, SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; OR
3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

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 document 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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

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

Programming interface information

This API descriptions 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 Presentation
Advanced Peer-to-Peer Networking
AFP
AIX
AnyNet
AS/400
BCOCA
C/400
COBOL/400
Common User Access
CUA
DB2
DB2 Universal Database
Distributed Relational Database Architecture
Domino
DPI
DRDA
Enterprise Storage Server
eServer
FlashCopy
GDDM
i5/OS
IBM
IBM (logo)
InfoColor
Infoprint
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
Lotus
Lotus Notes
MO:DCA
MVS
Net.Data
NetServer
Notes
OfficeVision
Operating System/2
Operating System/400
OS/2
OS/400
PartnerWorld
POWER5+
PowerPC
Print Services Facility
PrintManager
PROFS
RISC System/6000
RPG/400
RS/6000

SAA
SecureWay
SOM
System i
System i5
System Object Model
System/36
System/38
System/390
TotalStorage
VisualAge
WebSphere
xSeries
z/OS

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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, or 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