Object APIs (V5R2)

Table of Contents

        - Retrieve User Space (QUSRTVUS)
        - Retrieve User Space Attributes (QUSRUSAT)
- Object-related APIs
    - Change Library List (QLICHGLL)
    - Change Object Description (QLICOBJD)
    - Convert Type (QLICVTTP)
    - List Objects (QUSLOBJ)
    - Materialize Context (QusMaterializeContext)
    - Move Folder to ASP (QHSMMOVF)
    - Move Library to ASP (QHSMMOVL)
    - Open List of Objects (QGYOLOBJ)
    - Rename Object (QLIRNMO)
    - Retrieve Library Description (QLIRLIBD)
    - Retrieve Object Description (QUSROBJD)

Object APIs

The Object APIs create, manipulate, and delete user spaces, user indexes, and user queues. They send, receive, and clear entries on a data queue and retrieve data queue information. They also change, list, rename, and retrieve information about iSeries objects.

The Object APIs include:

- Data Queue APIs
- User Queue APIs
- User Index APIs
- User Space APIs
- Object-related APIs

APIs by category

Data Queue APIs

Data queues are a type of system object that you can create, to which one high-level language (HLL) program can send data, and from which another HLL program can receive data. The receiving program can be waiting for the data, or can receive the data later.

Before using a data queue, you must first create it using the Create Data Queue (CRTDTAQ) command.

For additional information, see Using Data Queue APIs.

The data queue APIs are:

- Clear Data Queue (QCLRDTAQ) clears entries from a data queue.
- Receive Data Queue (QRCVDTAQ) receives data from the specified data queue.
- Retrieve Data Queue Description (QMHQRDQD) retrieves information about a data queue.
- Retrieve Data Queue Message (QMHRDQM) retrieves an entry from a data queue without removing the entry.
- Send Data Queue (QSNDDTAQ) sends data to the specified data queue.

Using Data Queue APIs

The advantages of using data queues are:

- Using data queues frees a job from performing some work. If the job is an interactive job, the data queue APIs can provide better response time and decrease the size of the interactive program and its process activation group (PAG). This, in turn, can help overall system performance. For example, if several work station users enter a transaction that involves updating and adding to several files, the system can perform better if the interactive jobs submit the request for the transaction to a single batch processing job.

- Data queues are a fast means of asynchronous communication between two jobs. Using a data queue to send and receive data requires less system resource than using database files, message queues, or data areas to send and receive data.

- You can send to, receive from, and retrieve a description of a data queue in any HLL program. This is done by calling the Send to a Data Queue (QSNDDTAQ), Receive from Data Queue (QRCVDTAQ), Retrieve Data Queue Message (QMHRDQM), Clear Data Queue (QCLRDTAQ), and Retrieve Data Queue Description (QMHQRDQD) APIs.

- When receiving data from a data queue, you can set a time-out such that the job waits until an entry arrives on the data queue. This is different from using the EOFDLY parameter on the Override Database File (OVRDBF) command, which causes the job to be activated whenever the delay time ends.

- More than one job can receive data from the same data queue. This is an advantage in certain applications where the number of entries to be processed is greater than one job can handle within the desired performance restraints. For example, if several printers are available to print orders, several interactive jobs could send requests to a single data queue. A separate job for each printer could receive data from the data queue in first-in-first-out (FIFO), last-in-first-out (LIFO), or keyed-queue order.

- Data queues have the ability to attach a sender ID to each message being placed on the queue. The sender ID, an attribute of the data queue which is established when the queue is created, contains the qualified job name and current user profile.

Top | Object APIs | APIs by category

Clear Data Queue (QCLRDTAQ) API

Required Parameter Group:

1	Data queue name	Input	Char(10)
2	Library name	Input	Char(10)

Optional Parameter Group:

3	Key order	Input	Char(2)
4	Length of key data	Input	Packed(3,0)
5	Key data	Input	Char(*)
6	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Conditional; see Usage Notes.

The Clear Data Queue (QCLRDTAQ) API clears all data from the specified data queue, or clears messages that match the key specification from a keyed data queue.

If the data queue was created with the AUTORCL keyword on the Create Data Queue (CRTDTAQ) command set to *YES, when the queue is empty the storage allocated to the data queue will be reduced to the storage needed for the initial number of entries defined for the data queue.

Distributed data management (DDM) data queues are supported using this API. This means that you can use this API to clear a data queue that exists on a remote iSeries. Clearing messages by key is not supported for DDM data queues.

Authorities and Locks

Data Queue Authority

*OBJOPR and *READ

Data Queue Library Authority

*EXECUTE

Data Queue Lock

*EXCLRD

Required Parameter Group

Data queue name

INPUT; CHAR(10)

The name of the data queue being cleared.

Library name

INPUT; CHAR(10)

The name of the library where the data queue resides.

You can use these special values for the library name:

**LIBL* The library list

**CURLIB* The job's current library.

Optional Parameter Group

Key order

INPUT; CHAR(2)

The comparison criteria between the keys of messages on the data queue and the key data parameter.

Valid values are:

GT Greater than

LT Less than

NE Not equal

EQ Equal

GE Greater than or equal

LE Less than or equal

This parameter is ignored if the length of key data is zero. A value of blanks is recommended if the length of key data is zero.

For example, assume a keyed data queue contains these three entries:

Physical Entry	3-Character Key
1	GGG
2	XXX
3	AAA

If a key order of LT is specified with key data of XXX, entries 1 and 3 would be removed. If a key

order of EQ is specified with key data of XXX, entry 2 would be removed.

Length of key data

INPUT; PACKED(3,0)

The length of the key data parameter. If this parameter is specified, it must be zero for nonkeyed data queues. For keyed data queues it must be either zero or equal to the length specified on the KEYLEN parameter on the Create Data Queue (CRTDTAQ) command. If this parameter is not specified or is zero, all messages will be cleared from the data queue.

Key data

INPUT; CHAR(*)

The data to be used for selecting messages to be removed from the data queue.

Error code

I/O; CHAR(*)

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

Usage Notes

This API can be used in a multithreaded job to clear messages from a local data queue. It cannot be used in a job that allows multiple threads to clear messages from a DDM data queue.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9502 E	Key length must be zero for data queue &1 in &2.
CPF9503 E	Cannot lock data queue &1 in &2.
CPF9504 E	An invalid search order was specified.
CPF9506 E	Key length must be &3 for data queue &1 in &2.
CPF9507 E	Invalid key length specified.
CPF9510 E	Operation on DDM data queue &1 in &2 failed.
CPF9511 E	Function not supported for DDM data queue &1.
CPF9523 E	Data queue function not successful.

CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V1R3

Top | Object API categories | API by category

Receive Data Queue (QRCVDTAQ) API

Required Parameter Group:

1	Data queue name	Input	Char(10)
2	Library name	Input	Char(10)
3	Length of data	Output	Packed(5,0)
4	Data	Output	Char(*)
5	Wait time	Input	Packed(5,0)

Optional Parameter Group 1:

6	Key order	Input	Char(2)
7	Length of key data	Input	Packed(3,0)
8	Key data	I/O	Char(*)
9	Length of sender information	Input	Packed(3,0)
10	Sender information	Output	Char(*)

Optional Parameter Group 2:

11	Remove message	Input	Char(10)
12	Size of data receiver	Input	Packed(5,0)
13	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Conditional; see Usage Notes.

The Receive Data Queue (QRCVDTAQ) API receives data from the specified data queue.

When more than one program has a receive pending on a data queue at one time, a data entry sent to the data queue is received by only one of the programs. The program with the highest run priority receives the entry. The next entry sent to the queue is given to the job with the next highest priority.

If the data queue was created with the AUTORCL keyword on the Create Data Queue (CRTDTAQ) command set to *YES, when the queue is empty the storage allocated to the data queue will be reduced to the storage needed for the initial number of entries defined for the data queue.

Distributed data management (DDM) data queues are supported using this API. This means that you can use this API to receive a message from a data queue that exists on a remote iSeries. However, using this API to receive messages without removing them from the data queue is not supported for DDM data queues.

Authorities and Locks

Data Queue Authority

*OBJOPR and *READ

Data Queue Library Authority

*EXECUTE

Data Queue Lock

*EXCLRD

Internally, when a job uses API QSNDDTAQ (Send Data Queue), QRCVDTAQ (Receive Data Queue), QMHQRDQD (Retrieve Data Queue Description), or QMHRDQM (Retrieve Data Queue Message), a cache is created to allow faster access to the data queue. An entry in the cache means a user is authorized to the data queue. An entry is added to the cache when a user calling one of the APIs has the required authority to the data queue. An entry is also added to the cache when QSNDDTAQ is called to handle a journal entry for a data queue created with the sender ID attribute set to *YES, and the user requesting the the send function has the required authority to the current profile name in the sender ID information of the journal entry. The data in the cache is used until the job ends, so if you need to immediately change a user's authority to one of these objects, you may need to end that user's jobs.

Required Parameter Group

Data queue name

INPUT; CHAR(10)

The name of the data queue to receive the data from.

Library name

INPUT; CHAR(10)

The name of the library where the data queue resides.

You can use these special values for the library name:

**LIBL*	The library list
**CURLIB*	The job's current library.

Note: To improve data queue performance, the data queue APIs remember addressing information for the last data queues used. This occurs when a specific (not *LIBL or *CURLIB) value is provided for the library name, » and the data queue is located in the system auxiliary storage pool (ASP number 1) or a basic user ASP (ASP numbers 2-32). The addressing information for data queues located in independent ASPs is not saved. «

Because the addressing information is saved, users of this API should be aware of the following scenarios.

Scenario 1

If, a job references a library-specific data queue, the data queue is moved using the Move Object (MOVOBJ) command or renamed using the Rename Object (RNMOBJ) command, and a new data

queue is created with the same name and library as the data queue that was renamed or moved, then, the job continues to reference the original data queue, not the newly created data queue.

Scenario 2

If, a job references a library-specific distributed data management (DDM) data queue, the DDM data queue is moved using the Move Object (MOVOBJ) command or renamed using the Rename Object (RNMOBJ) command, and a new data queue is created with the same name and library as the DDM data queue that was renamed or moved, the job continues to reference the original DDM data queue, not the newly created data queue.

Scenario 3

If, a job references a DDM data queue, which starts a DDM target job (DDM conversation) on a remote system that references a library-specific data queue, the data queue on the remote system is moved using the Move Object (MOVOBJ) command or renamed using the Rename Object (RNMOBJ) command, and on the remote system, a new data queue is created with the same name and library as the data queue that was renamed or moved, then, the DDM target job continues to reference the original data queue on the remote system, not the newly created data queue, only when the same DDM target job is used for the subsequent data queue operation. If a new DDM target job is used for the subsequent data queue operation, then the newly created data queue will be used on the remote system.

Note: For more information on creating DDM data queues and on DDM target jobs, see Distributed Data Management in the iSeries Information Center.

Length of data

OUTPUT; PACKED(5,0)

The number of characters received from the data queue. If a time out occurs and no data is received from the data queue, this field is set to zero. The value of this field will never exceed the value specified for the MAXLEN parameter on the Create Data Queue (CRTDTAQ) command.

If the size of the data receiver variable is specified, the data received from the data queue will be truncated if the message is longer than the size of the data receiver variable. The value of this field will be set to the actual length of the data received before it is truncated.

Data

OUTPUT; CHAR(*)

A field of at least the length of the value specified for the MAXLEN parameter on the Create Data Queue (CRTDTAQ) command. This field contains the data received from the data queue.

Note: If the length of this field is larger than the size of the message received, only the number of characters (beginning from the left) as defined by the message received from the data queue are changed. If the length of this field is smaller than the value specified for the MAXLEN parameter on the Create Data Queue (CRTDTAQ) command, and the actual length of this field is not specified in the size of data receiver parameter, unexpected results can occur.

If the length of this field is specified in the size of data receiver parameter, the data received will be truncated if it is longer than the size specified.

Wait time

INPUT; PACKED(5,0)

The amount of time to wait if no entries exist on the data queue.

When no entries are on the data queue, the wait time parameter (in seconds) specifies the

following:

< 0 Waits forever.

0 Continue processing immediately. If no entry exists, the call completes immediately with the length of data parameter set to zero.

> 0 The number of seconds to wait. The maximum is 99999 which allows a wait time of approximately 28 hours.

Notes: If the wait time value is less than or equal to 2, the job does not leave the activity level (for 2 seconds). This is described as a short wait. For more details on activity levels and implementation applications, see the Work Management book on the V5R1 Supplemental Manuals Web site.

»If a wait time is specified when receiving an entry from a data queue located in an independent auxiliary storage pool (ASP), and the independent ASP is varied off:

- If *YES is specified or defaulted for the remove message parameter, QRCVDTAQ will end with an exception, and no data will be returned.
- If *NO is specified for the remove message parameter, QRCVDTAQ will wait for the specified wait time, and no data will be received, even if the independent ASP is varied on again and a message is sent to the data queue.«

Optional Parameter Group 1

Key order

INPUT; CHAR(2)

The comparison criteria between the keys of messages on the data queue and the key data parameter. When the system searches for the requested key, the entries are searched in ascending order from the lowest value key to the highest value key until a match is found. If there are entries with duplicate keys, the entry that was put on the queue first is received.

Valid values are:

GT Greater than

LT Less than

NE Not equal

EQ Equal

GE Greater than or equal

LE Less than or equal

This parameter is ignored if the length of key data is zero. A value of blanks is recommended if the length of key data is zero.

For example, assume a keyed data queue contains these three entries:

Physical Entry 3-Character Key

1	GGG
2	XXX
3	AAA

If a key order of LE is specified with key data of XXX, entry 3 would be received. If the same values were specified on a subsequent request, entry 1 would be received.

Length of key data

INPUT; PACKED(3,0)

The length of the key data parameter. If this parameter is specified, it must be zero for nonkeyed data queues. For keyed data queues it must be equal to the length specified on the KEYLEN parameter on the Create Data Queue (CRTDTAQ) command.

Key data

I/O; CHAR(*)

The data to be used for receiving a message from the data queue. The key of the received message is also returned in this field. It may be different than the key specified to search for. For example, if the key data parameter is set to AA on input with the key order parameter set to GE (greater than or equal to), the key of the record that is actually received could be AB or anything else greater than or equal to AA. The key data parameter is set to the actual key of the received data when the API returns.

Length of sender information

INPUT; PACKED(3,0)

The length of the sender identification parameter.

Valid values are:

0	No sender information is returned.
8	Returns only the bytes returned and bytes available fields of the sender information.
> 8	Return as much sender information as the length allows.

Sender information

OUTPUT; CHAR(*)

The sender ID information associated with the received message.

Format of Sender Information

The format and content of the sender information returned is shown in the following table. For a detailed description of each field, see Field Descriptions.

Note: On the CRTDTAQ command, the SENDERID parameter defaults to *NO. To include the sender ID for each data queue entry, the SENDERID parameter must be *YES when the data queue is created.

Offset		

Dec	Hex	Type	Field
0	0	PACKED(7,0)	Bytes returned
4	4	PACKED(7,0)	Bytes available
8	8	CHAR(10)	Job name
18	12	CHAR(10)	User profile name
28	1C	CHAR(6)	Job number
34	22	CHAR(10)	Senders current user profile name.
Note: The last four fields, together, combine to make up the sender ID.			

Field Descriptions

Bytes available. The number of bytes of data available to be returned. All available data is returned if enough space is provided.

Bytes returned. The number of bytes of data returned.

Job name. The name of the job that sent the message.

Job number. The job number of the job that sent the message.

Senders current user profile name. The current user profile name of the job that sent the message.

User profile name. The user profile name of the job that sent the message.

Optional Parameter Group 2

Remove message

INPUT; CHAR(10)

Whether the message is to be removed from the data queue when it is received.

Valid values are:

**YES* The message is removed from the data queue. This is the default value if this parameter is not specified.

**NO* The message is not removed from the data queue.

Size of data receiver

INPUT; PACKED(5,0)

The size of the area to contain the data received from the data queue. If a value of 0 is specified for this parameter, no data will be returned. If a size greater than 0 is specified, the data will be copied into the receiver up to the specified length. If the available data is longer than the length specified, it will be truncated.

If this parameter is not specified, the entire message will be copied into the receiver variable.

Error code

I/O; CHAR(*)

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

Usage Notes

This API can be used in a multithreaded job to receive messages from a local data queue. It cannot be used in a job that allows multiple threads to receive messages from a DDM data queue.

Application queueing time and resource usage time for data queue usage are recorded only for messages that are received in the initial thread of a job.

Error Messages

Message ID	Error Message Text
CPF2207 E	Not authorized to use object &1 in library &3 type *&2.
CPF24B4 E	Severe error while addressing parameter list.
CPF2472 E	Invalid wait time specified.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9501 E	Data queue &1 in &2 requires a key value.
CPF9502 E	Key length must be zero for data queue &1 in &2.
CPF9503 E	Cannot lock data queue &1 in &2.
CPF9504 E	An invalid search order was specified.
CPF9505 E	Sender ID length value is not valid.
CPF9506 E	Key length must be &3 for data queue &1 in &2.
CPF9507 E	Invalid key length specified.
CPF9508 E	Invalid sender ID length specified.
CPF9509 E	Space access error.
CPF9510 E	Operation on DDM data queue &1 in &2 failed.
CPF9511 E	Function not supported for DDM data queue &1.
CPF9514 E	Value for data length parameter not valid.
CPF9515 E	Value for remove message parameter not valid.

CPF9523 E	Data queue function not successful.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
»CPF9803 D	Cannot allocate object &2 in library &3.«
CPF9805 E	Object &2 in library &3 destroyed.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V1R1

Top | Object API categories | API by category

Retrieve Data Queue Description (QMHQRDQD) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Qualified data queue name	Input	Char(20)

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve Data Queue Description (QMHQRDQD) API retrieves the description and attributes of a data queue. Examples include the number of entries currently on the data queue, the text description of the data queue, whether the queue includes sender ID information, and whether the data queue is keyed.

The attributes of a distributed data management (DDM) data queue can be retrieved with this API.

Authorities and Locks

Data Queue Authority

*OBJOPR and *READ

Data Queue Library Authority

*EXECUTE

Data Queue Lock

*EXCLRD

Internally, when a job uses API QSNDDTAQ (Send Data Queue), QRCVDTAQ (Receive Data Queue), QMHQRDQD (Retrieve Data Queue Description), or QMHRDQM (Retrieve Data Queue Message), a cache is created to allow faster access to the data queue. An entry in the cache means a user is authorized to the data queue. An entry is added to the cache when a user calling one of the APIs has the required authority to the data queue. An entry is also added to the cache when QSNDDTAQ is called to handle a journal entry for a data queue created with the sender ID attribute set to *YES, and the user requesting the the send function has the required authority to the current profile name in the sender ID information of the journal entry. The data in the cache is used until the job ends, so if you need to immediately change a user's authority to one of these objects, you may need to end that user's jobs.

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 provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes.

Format name

INPUT; CHAR(8)

The format of the data queue description to be returned.

The valid format names are:

RDQD0100 Basic data queue description.

RDQD0200 DDM data queue description. This is valid for DDM data queues only.

Qualified data queue name

INPUT; CHAR(20)

The data queue whose description is to be returned. The first 10 characters contain the data queue name, and the second 10 characters contain the data queue library name.

You can use these special values for the library name:

**CURLIB* The job's current library

**LIBL* The library list

Note: To improve data queue performance, the data queue APIs remember addressing information for the last data queues used. This occurs when a specific (not *LIBL or *CURLIB) value is provided for the library name, » and the data queue is located in the system auxiliary storage pool (ASP number 1) or a basic user ASP (ASP numbers 2-32). The addressing information for data queues located in independent ASPs is not saved. «

Because the addressing information is saved, users of this API should be aware of the following scenario:

If, a job references a data queue the data queue is moved using the Move Object (MOVOBJ) command or renamed using the Rename Object (RNMOBJ) command, and a new data queue is created with the same name and library as the data queue that was renamed or moved, then, the job continues to reference the original data queue, not the newly created data queue.

The actual name of the data queue and the library in which it is found are returned by this API.

RDQD0100 Format

The following table shows the information placed in the receiver variable parameter for the RDQD0100 format. For a detailed description of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Message length
12	C	BINARY(4)	Key length
16	10	CHAR(1)	Sequence
17	11	CHAR(1)	Include sender ID
18	12	CHAR(1)	Force indicator
19	13	CHAR(50)	Text description
69	45	CHAR(1)	Type of data queue
70	46	CHAR(1)	Automatic Reclaim
71	47	CHAR(1)	Reserved
72	48	BINARY(4)	Number of messages
76	4C	BINARY(4)	Number of entries currently allocated
80	50	CHAR(10)	Data queue name used
90	5A	CHAR(10)	Data queue library used
100	64	BINARY(4)	Maximum number of entries allowed
104	68	BINARY(4)	Initial number of entries

RDQD0200 Format

The following table shows the information placed in the receiver variable parameter for the RDQD0200 format. For a detailed description of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	APPC device description
18	12	CHAR(8)	Mode
26	1A	CHAR(8)	Remote location name
34	22	CHAR(8)	Local location name
42	2A	CHAR(8)	Remote network identifier
50	32	CHAR(10)	Remote data queue name
60	3C	CHAR(10)	Remote data queue library name

70	46	CHAR(10)	Data queue name used
80	50	CHAR(10)	Data queue library used

Field Descriptions

APPC device description. The name of the APPC device description on the source system that is used with this DDM data queue. The special value *LOC can be returned. This is the name that was specified on the DEV parameter of the CRTDTAQ command.

Automatic reclaim. Whether or not the data queue has the amount of storage allocated for the queue reclaimed when the queue is empty.

Possible values returned are:

0 Storage is not reclaimed.

1 Storage is reclaimed when the queue is empty. The amount of storage allocated will be set to the initial number of entries.

This will be blank for a DDM data queue.

Bytes available. The number of bytes of data available to be returned. All available data is returned if enough space is provided.

Bytes returned. The number of bytes of data returned.

Data queue library used. The library in which the data queue is found. If *LIBL or *CURLIB is specified for the library name, this field is the actual name of the library in which the data queue was found. If a specific library (not *LIBL or *CURLIB) is specified, and the data queue is moved from that library to a different library after this job first accessed the data queue, this will be set to the name of the library in which the data queue currently exists.

Data queue name used. The name of the data queue. This will be the same as the name specified unless the data queue was renamed after this job first accessed the data queue.

Force indicator. Whether or not the data queue is forced to auxiliary storage when entries are sent or received for the specified data queue.

Possible values returned are:

Y The data queue is forced to auxiliary storage after entries are sent or received.

N The data queue is not forced to auxiliary storage after entries are sent or received.

This will be blank for a DDM data queue.

Include sender ID. If the queue was created to include the sender ID with sent messages.

Possible values returned are:

Y The sender ID is included when data is sent to the data queue.

N The sender ID is not included when data is sent to the data queue.

This will be blank for a DDM data queue.

Initial number of entries. The number of messages that will fit into the storage allocated for the data queue when it is created or when it is automatically reclaimed. This will be 0 for a DDM data queue.

Key length. If the specified data queue was created as a keyed type, this field contains the length, in bytes, of the message reference key. Values range from 1 to 256. If the specified queue is not a keyed queue or is a DDM data queue, the value is 0.

Local location name. The name of the local location. The special values *LOC and *NETATR can be returned. This is the name that was specified on the LCLLOCNAME parameter of the CRTDTAQ command.

Maximum number of entries allowed. The maximum number of messages that will fit into the data queue when it is full. This will be 0 for a DDM data queue.

Message length. The maximum length allowed for messages. The is the value that was specified with the MAXLEN keyword on the CRTDTAQ command. This will be 0 for a DDM data queue.

Mode. The mode name used with the remote location name to communicate with the target system. The special value *NETATR can be returned. This is the name that was specified on the MODE parameter of the CRTDTAQ command.

Number of entries currently allocated. The number of entries that will fit into the data queue before it is extended. When the queue is extended, additional storage is allocated for the queue. The data queue can be extended until it reaches the value for the maximum number of entries allowed. This will be 0 for a DDM data queue.

Number of messages. The number of messages currently on the data queue. This will be 0 for a DDM data queue.

Remote data queue library name. The name of the library for the remote data queue on the target system. The special values *LIBL and *CURLIB can be returned. This is the data queue name that was specified on the RMTDTAQ parameter of the CRTDTAQ command.

Remote data queue name. The name of the remote data queue on the target system. This is the data queue name that was specified on the RMTDTAQ parameter of the CRTDTAQ command.

Remote location name. The name of the remote location that is used with this object. This is the name that was specified on the RMTLOCNAME parameter of the CRTDTAQ command.

Remote network identifier. The remote network identifier in which the remote location used to communcate with the target system. The special values *LOC, *NETATR, and *NONE can be returned. This is the name that was specified on the RMTNETID parameter of the CRTDTAQ command.

Reserved. An unused field.

Sequence. The sequence in which messages can be removed from the queue.

Possible values returned are:

F First-in first-out

K Keyed

L Last-in first-out

This will be blank for a DDM data queue.

Text description. The text description of the data queue. The field contains blanks if no text description was specified when the data queue was created.

Type of data queue.

This will be set to one of the following values:

0 The data queue is a standard data queue.

1 The data queue is a DDM data queue.

Error Messages

Message ID	Error Message Text
CPF2150 E	Object information function failed.
CPF2151 E	Operation failed for &2 in &1 type *&3.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9503 E	Cannot lock data queue &1 in &2.
CPF9509 E	Space access error.
CPF9516 E	Format &1 not allowed for data queue.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

Retrieve Data Queue Message (QMHRDQM) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Qualified data queue name	Input	Char(20)
5	Message selection information	Input	Char(*)
6	Length of message selection information	Input	Binary(4)
7	Message selection information format name	Input	Char(8)
8	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve Data Queue Message (QMHRDQM) API retrieves one or more messages from a data queue.

The QMHRDQM API allows the retrieval of multiple messages per call. The message selection information parameter allows you to have some control over which messages are returned. The QMHRDQM API can be used to retrieve the following:

- The first or last message of a data queue
- All messages of a data queue
- Selected messages from a keyed data queue

The QMHRDQM API is similar in function to the QRCVDTAQ API. However, the QRCVDTAQ API can remove the received message from the data queue; QMHRDQM API does **not** remove received messages.

Distributed data management (DDM) data queues are not supported using this API.

Authorities and Locks

Data Queue Authority

*OBJOPR and *READ

Data Queue Library Authority

*EXECUTE

Data Queue Lock

*EXCLRD

Internally, when a job uses API QSNDDTAQ (Send Data Queue), QRCVDTAQ (Receive Data Queue), QMHQRDQD (Retrieve Data Queue Description), or QMHRDQM (Retrieve Data Queue Message), a cache is created to allow faster access to the data queue. An entry in the cache means a user is authorized to

the data queue. An entry is added to the cache when a user calling one of the APIs has the required authority to the data queue. An entry is also added to the cache when QSNDDTAQ is called to handle a journal entry for a data queue created with the sender ID attribute set to *YES, and the user requesting the the send function has the required authority to the current profile name in the sender ID information of the journal entry. The data in the cache is used until the job ends, so if you need to immediately change a user's authority to one of these objects, you may need to end that user's jobs.

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 provided. The length of receiver variable parameter may be specified up to the size of the receiver variable specified in the user program. If the length of receiver variable parameter specified is larger than the allocated size of the receiver variable specified in the user program, the results are not predictable. The minimum length is 8 bytes

Format name

INPUT; CHAR(8)

The format of the data to be placed in the receiver variable.

The valid format names are:

RDQM0100 Retrieved data queue message(s) information.

RDQM0200 Retrieved data queue message(s) information.

The RDQM0200 format differs from RDQM0100 in that when the user requests to retrieve the number of bytes equal to the maximum message entry size specified, this returns the length and data of the message entry that was enqueued instead of the maximum length and data contained in the data queue entry. Therefore, the repeating fields at the end of format RDQM0200 could require less space than format RDQM0100, which uses the same amount of space for each entry retrieved from the queue.

Qualified data queue name

INPUT; CHAR(20)

The data queue whose description is to be returned. The first 10 characters contain the data queue name, and the second 10 characters contain the data queue library name.

You can use these special values for the library name:

*CURLIB The job's current library

*LIBL The library list

Note: To improve data queue performance, the data queue APIs remember addressing information for the last data queues used. When this occurs, a specific (not *LIBL or *CURLIB) value is provided for the library name, » and the data queue is located in the system auxiliary storage pool (ASP number 1) or a basic user ASP (ASP numbers 2-32). The addressing information for data queues located in independent ASPs is not saved. «

Because the addressing information is saved, users of this API should be aware of the following scenario:

If, a job references a data queue, the data queue is moved using the Move Object (MOVOBJ) command or renamed using the Rename Object (RNMOBJ) command, and a new data queue is created with the same name and library as the data queue that was renamed or moved, then, the job continues to reference the original data queue, not the newly created data queue.

Message selection information

INPUT; CHAR(*)

Identifies which message (or messages) you want to retrieve. The layout of this parameter is determined by the value of the message selection information format name.

Length of message selection information

INPUT; BINARY(4)

The length of the message selection information parameter. This must be 8 bytes for RDQS0100 and 16 bytes plus the size of the key for RDQS0200.

Message selection information format name

INPUT; CHAR(8)

The format of the message selection information parameter.

The following format names can be used:

RDQS0100 Format to select messages when using nonkeyed data queues.

RDQS0200 Format to select messages when using keyed data queues.

Error code

I/O; CHAR(*)

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

RDQM0100 Format

The following table lists the fields returned in the RDQM0100 format of the receiver variable parameter. For a detailed description of each field, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available

8	8	BINARY(4)	Number of messages returned
12	C	BINARY(4)	Number of messages available
16	10	BINARY (4)	Message key length returned
20	14	BINARY(4)	Message key length available
24	18	BINARY(4)	Maximum message text length requested
28	1C	BINARY(4)	Maximum message text length available
32	20	BINARY(4)	Entry length returned
36	24	BINARY(4)	Entry length available
40	28	BINARY(4)	Offset to first message entry
44	2C	CHAR(10)	Actual data queue library name
54	36	CHAR(*)	Reserved
These fields repeat for each message retrieved.		BINARY(4)	Offset to next message entry
		CHAR(8)	Message enqueue date and time
		CHAR(*)	Message key
		CHAR(*)	Message text
		CHAR(*)	Reserved

RDQM0200 Format

The following table lists the fields returned in the RDQM0200 format of the receiver variable parameter. For a detailed description of each field, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Number of messages returned
12	C	BINARY(4)	Number of messages available
16	10	BINARY (4)	Message key length returned
20	14	BINARY(4)	Message key length available
24	18	BINARY(4)	Maximum message text length requested
28	1C	BINARY(4)	Maximum message text length available
32	20	CHAR(8)	Reserved
40	28	BINARY(4)	Offset to first message entry
44	2C	CHAR(10)	Actual data queue library name
54	36	CHAR(*)	Reserved
These fields repeat for each message retrieved.		BINARY(4)	Offset to next message entry
		CHAR(8)	Message enqueue date and time
		BINARY(4)	Enqueued message entry length
		CHAR(*)	Message key
		CHAR(*)	Message text
		CHAR(*)	Reserved

RDQS0100 Format

The following table describes the RDQS0100 format of the Message selection information parameter. This format is used with data queues when selection with keys is not necessary. This format cannot be used with keyed data queues. To retrieve messages using keys and key search order, use format RDQS0200. For a detailed description of each field, see Field Descriptions.

For example, to retrieve the first 10 bytes of the last entry in a data queue specify the following: Parameter seven would contain a format name of 'RDQS0100' and parameter five would consist of a Selection type of 'L', the Reserved field would be blanks, and Number of message text bytes to retrieve would be 10.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(1)	Selection type
1	1	CHAR(3)	Reserved
4	4	BINARY(4)	Number of message text bytes to retrieve

RDQS0200 Format

The following table describes the RDQS0200 format of the Message selection information parameter. This format is used to retrieve messages from data queues when selection with keys is necessary. When using this format, all messages satisfying the key search order are returned. For a detailed description of each field, see Field Descriptions.

Note: This format is valid only if the queue was created as a keyed data queue.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(1)	Selection type
1	1	CHAR(2)	Key search order
3	3	CHAR(1)	Reserved
4	4	BINARY(4)	Number of message text bytes to retrieve
8	8	BINARY(4)	Number of message key bytes to retrieve
12	C	BINARY(4)	Length of Key
16	10	CHAR(*)	Key

Field Descriptions

Actual data queue library name. The library in which the data queue was found. This name is found by searching the library list (*LIBL) or the current library (*CURLIB). If the data queue is in a library other than your current library or library list, it will not be found.

Bytes available. The number of bytes of data available to be returned. All available data is returned if enough space is provided.

Bytes returned. The number of bytes of data returned.

Enqueued message entry length. The number of bytes specified for the message entry length when the entry was placed on the data queue. For a data queue created with SENDERID(*YES), this length is the message entry length specified plus 36 bytes for the sender ID. This is the number of bytes returned in the message text field unless the number of message text bytes to retrieve field is less than this value. In that case, the maximum message text length requested is returned in the message text field.

Entry length available. The total number of bytes available to be retrieved for each message entry.

Entry length returned. The number of bytes retrieved for each message entry.

Key. The key field to be compared with the actual keys of the messages on the data queue.

Key search order. A relational operator specifying the comparison criteria between the message key specified in the RDQS0200 format and the actual keys of messages in the data queue.

Valid values are:

GT	All messages with a key greater than that specified in the key field are to be returned.
LT	All messages with a key less than that specified in the key field are to be returned.
NE	All messages with a key not equal to that specified in the key field are to be returned.
EQ	All messages with a key equal to that specified in the key field are to be returned.
GE	All messages with a key greater than or equal to that specified in the key field are to be returned.
LE	All messages with a key less than or equal to that specified in the key field are to be returned.

Length of key. The length of the data provided in the key field. This must be a value from 1 through 256.

Maximum message text length available. The maximum message entry size (in bytes) specified when the data queue was created. For a data queue created with SENDERID(*YES), this length is the maximum entry size plus 36 bytes for the sender ID.

Maximum message text length requested. The value specified in the message selection format (RDQS0100 or RDQS0200) for the number of message text bytes to retrieve.

Message enqueue date and time. The date and time that the message was placed on the data queue. Its format is a system time stamp (*DTS). The Convert Date and Time Format (QWCCVTDT) API can be used to convert this time stamp to a character format.

Message key. The key of the message.

Message key length available. The size (in bytes) of the key at the creation time of the data queue.

Message key length returned. The number of bytes retrieved in the message key field.

Message text. The text of the message. For the RDQM0100 format, the number of bytes of message text returned is the maximum message text length requested. For the RDQM0200 format, the number of bytes of message text returned is the minimum of the maximum message text length requested field or the enqueued message entry length field. For example, with RDQM0200, if the maximum message text length requested is less than the enqueued message entry length, some text of the data queue entry would not be

returned.

Number of message key bytes to retrieve. The number of message key bytes to return for each data queue entry. The maximum value allowed is 256. If the number of bytes requested exceeds the actual message key length, the key is padded with binary zeros. If the number of message key bytes requested is less than the actual key, the key is truncated.

Number of message text bytes to retrieve. The number of message text bytes to return for each data queue entry. The maximum value allowed is 65536. The maximum message text length returned field is equal to this value. If the number of bytes requested exceeds the actual message text length, the text is padded with binary zeros if the RDQM0100 format is used. If the RQDM0200 format is used, and the actual message text length is less than the number of bytes requested, the number of bytes returned is the actual message length, and no padding is done. If the number of message text bytes is less than the actual text, the message text is truncated.

Number of messages available. The number of messages on the data queue that satisfy the search criteria specified in the Message selection information parameter.

Number of messages returned. The number of messages retrieved.

Offset to first message entry. The offset at which the first message entry begins. If this value is 0, there is no message available.

Offset to next message entry. The offset to the next message entry. If this value is 0, there are no more messages returned.

Reserved. An unused field.

Selection type. Selection type depends on the format used.

For the RDQS0100 format, valid values are:

A All messages are to be returned in the order based on the type of data queue. FIFO queues are returned in FIFO order, LIFO queues are returned in LIFO order and keyed queues are returned in ascending key order.

F The first message is to be returned

L The last message is to be returned

R All messages are to be returned in reverse order of the type of data queue. For example, LIFO queues are returned in FIFO order.

For the RDQS0200 format, valid values are:

K Messages meeting the key criteria are to be returned

Error Messages

Message ID	Error Message Text
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.

CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9503 E	Cannot lock data queue &1 in &2.
CPF9504 E	An invalid search order was specified.
CPF9509 E	Space access error.
CPF9511 E	Function not supported for DDM data queue &1.
CPF950B E	The specified selection type is not valid.
CPF950C E	The specified retrieve length is not valid.
CPF950D E	The specified message selection template length is not valid.
CPF950E E	The data queue is not a keyed data queue.
CPF950F E	The specified key length is not valid.
CPF9519 E	Internal program error occurred.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R3

Top | Object API categories | API by category

Send Data Queue (QSNDDTAQ) API

Required Parameter Group:

1	Data queue name	Input	Char(10)
2	Library name	Input	Char(10)
3	Length of data	Input	Packed(5,0)
4	Data	Input	Char(*)

Optional Parameter Group 1:

5	Length of key data	Input	Packed(3,0)
6	Key data	Input	Char(*)

Optional Parameter Group 2:

7	Asynchronous request	Input	Char(10)

Optional Parameter Group 3:

8	Data is from a journal entry	Input	Char(10)

Default Public Authority: *USE

Threadsafe: Conditional; see Usage Notes.

The Send Data Queue (QSNDDTAQ) API sends data to the specified data queue. When an entry is sent to a standard data queue, the storage allocated for each entry will be the value specified for the maximum entry length on the Create Data Queue (CRTDTAQ) command.

Distributed data management (DDM) data queues are supported using this API. This means that you can use this API to send data to a data queue that exists on a remote iSeries.

Authorities and Locks

Data Queue Authority

*OBJOPR and *ADD

Data Queue Library Authority

*EXECUTE

When using optional parameter group 3 and the journal entry was deposited for a data queue with the

*sender ID attribute set to *YES, authority to the current profile in the sender ID of the journal entry*

*USE

Data Queue Lock

*EXCLRD

Internally, when a job uses API QSNDDTAQ (Send Data Queue), QRCVDTAQ (Receive Data Queue), QMHQRDQD (Retrieve Data Queue Description), or QMHRDQM (Retrieve Data Queue Message), a cache is created to allow faster access to the data queue. An entry in the cache means a user is authorized to the data queue. An entry is added to the cache when a user calling one of the APIs has the required authority to the data queue. An entry is also added to the cache when QSNDDTAQ is called to handle a journal entry for a data queue created with the sender ID attribute set to *YES, and the user requesting the the send function has the required authority to the current profile name in the sender ID information of the journal entry. The data in the cache is used until the job ends, so if you need to immediately change a user's authority to one of these objects, you may need to end that user's jobs.

Required Parameter Group

Data queue name

INPUT; CHAR(10)

The name of the data queue to send the data to.

Library name

INPUT; CHAR(10)

The name of the library where the data queue resides.

You can use these special values for the library name:

**LIBL* The library list

**CURLIB* The job's current library.

Note: To improve data queue performance, the data queue APIs remember addressing information for the last data queues used. This occurs when a specific (not *LIBL or *CURLIB) value is provided for the library name, » and the data queue is located in the system auxiliary storage pool (ASP number 1) or a basic user ASP (ASP numbers 2-32). The addressing information for data queues located in independent ASPs is not saved. «

Because the addressing information is saved, users of this API should be aware of the following scenarios.

Scenario 1

If, a job references a library-specific data queue, the data queue is moved using the Move Object (MOVOBJ) command or renamed using the Rename Object (RNMOBJ) command, and a new data queue is created with the same name and library as the data queue that was renamed or moved, then, the job continues to reference the original data queue, not the newly created data queue.

Scenario 2

If, a job references a library-specific distributed data management (DDM) data queue, the DDM

data queue is moved using the Move Object (MOVOBJ) command or renamed using the Rename Object (RNMOBJ) command, and a new data queue is created with the same name and library as the DDM data queue that was renamed or moved, then, the job continues to reference the original DDM data queue, not the newly created data queue.

Scenario 3

If, a job references a DDM data queue, which starts a DDM target job (DDM conversation) on a remote system that references a library-specific data queue, the data queue on the remote system is moved using the Move Object (MOVOBJ) command or renamed using the Rename Object (RNMOBJ) command, and on the remote system, a new data queue is created with the same name and library as the data queue that was renamed or moved, then, the DDM target job continues to reference the original data queue on the remote system, not the newly created data queue, only when the same DDM target job is used for the subsequent data queue operation. If a new DDM target job is used for the subsequent data queue operation, then the newly created data queue will be used on the remote system.

Note: For more information on creating DDM data queues and on DDM target jobs, see Distributed Data Management in the iSeries Information Center.

Length of data

INPUT; PACKED(5,0)

The number of characters to be sent to the data queue.

Note: An error occurs if the value specified is greater than the length specified by the maximum lenght (MAXLEN) parameter on the Create Data Queue (CRTDTAQ) command, unless optional parameter group 3 is specified. With optional parameter group 3, the length of the data provided in the journal entry should be specified, and it could be longer than the maximum entry length. This will be handled appropriately by the API.

Data

INPUT; CHAR(*)

The data to be sent to the data queue.

Note: If the length of this field is larger than the length of data parameter, only the number of characters (beginning from the left) as defined by the length of data parameter are sent to the data queue. If the length of this variable is smaller than the length of data parameter, unexpected results can occur.

Optional Parameter Group 1

Length of key data

INPUT; PACKED(3,0)

The number of characters in the key data parameter.

Note: The maximum value is the value that is specified on the KEYLEN parameter on the Create Data Queue (CRTDTAQ) command.

Key data

INPUT; CHAR(*)

The data sent to the data queue. This value must be at least as long as the value specified in the

length of key data parameter; otherwise, unexpected results can occur.

Optional Parameter Group 2

Asynchronous request

INPUT; CHAR(10)

Whether the send data queue request to a DDM data queue should be processed asynchronously. This parameter only applies to DDM data queues. Valid values are *YES and *NO. An error will occur if *YES is specified for a non-DDM data queue. If the value *YES is specified for the asynchronous request parameter for a DDM data queue and an error occurs on the operation, the error will not be detected until the next time the data queue is accessed.

Note: If this parameter is specified, the key length and key data parameters must also be specified even if they are not applicable. In that event, the key length should be zero and blanks should be specified for the key data.

Optional Parameter Group 3

Data is from a journal entry

INPUT; CHAR(10)

Indicate whether the data, in parameter four, came from a journal entry. This parameter only applies to a non-DDM data queue. Valid values are *YES and *NO.

When using this paramter the data queue needs to have the same attributes as the journaled data queue. To ensure the attributes of the data queue are the same as the journaled data queue, save the journaled data queue and restore it to the system where this API will be called with optional parameter group 3 specified. Refer to Journal management for information on journaling.

Note: When using this parameter, the length of data in parameter three should be obtained from the journal entry. The length could be longer than the maximum entry length of the data queue and this will be handled appropriately by the API, an error will not be issued.

Note: If this parameter is specified, the key length, key data, and asynchronous request parameters must also be specified even if they are not applicable. In that event, the key length should be zero, key data should be blanks and *NO should be specified for the asynchronous request.

Usage Notes

This API can be used in a multithreaded job to send entries to a non-DDM data queue. It cannot be used in a job that allows multiple threads to send entries to a DDM data queue.

Error Messages

Message ID	Error Message Text
CPF2207 E	Not authorized to use object &1 in library &3 type *&2.
CPF24B4 E	Severe error while addressing parameter list.
CPF2498 E	Invalid length. MAXLEN for data queue &1 in &2 is &3.
CPF3C90 E	Literal value cannot be changed.
CPF6565 E	User profile storage limit exceeded.
CPF9501 E	Data queue &1 in &2 requires a key value.
CPF9502 E	Key length must be zero for data queue &1 in &2.
CPF9503 E	Cannot lock data queue &1 in &2.
CPF9506 E	Key length must be &3 for data queue &1 in &2.
CPF9507 E	Invalid key length specified.
CPF9509 E	Space access error.
CPF950A E	Storage limit exceeded for data queue &1 in &2.
CPF9510 E	Operation on DDM data queue &1 in &2 failed.
CPF9512 E	Invalid asynchronous request.
CPF9513 E	Asynchronous request must be *NO for data queue &1 in &2.
CPF9520 E	Value for data is from a journal entry is not valid.
CPF9521 E	Entry not sent. Data queue attributes do not match.
CPF9522 E	Entry not sent. Data from a journal entry is not valid.
CPF9523 E	Data queue function not successful.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 D	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V1R1

Top | Object APIs | APIs by category

User Queue APIs

The user queue APIs let you create and delete user queues. **User queues** are permanent objects with an object type of *USRQ. They provide a way for one or more processes to communicate asynchronously.

For additional information, see Using User Queue APIs.

The user queue APIs are:

- Create User Queue (QUSCRTUQ) creates a user queue in either the user domain or the system domain, based on the input parameters.
- Delete User Queue (QUSDLTUQ) deletes user queues created with the Create User Queue (QUSCRTUQ) API.

Using User Queue APIs

You can use user queues to:

- Communicate between two processes asynchronously.
- Store data in arrival sequence for later use.
- Contain keyed messages.
- Create a batch machine. (For an example, see Creating a Batch Machine in the API Examples.)
- Permit better performance than the data queue interface.

You can save and restore a user queue; however, you can save or restore its definition only. You cannot save or restore the messages in it. You cannot restore a user queue if a user queue with the same name already exists in the library. You must provide programs to use this object type to enqueue and dequeue messages.

In addition to the user queue APIs, you can work with user queues through the following:

- ILE C programming language
- Delete User Queue (DLTUSRQ) command
- Machine interface (MI) instructions

For details about MI instructions, refer to the iSeries Machine Interface Instructions, which provides detailed descriptions of the iSeries machine interface instruction fields and the formats of those fields. For details about the DLTUSRQ command, see the Control Language topic.

Create User Queue (QUSCRTUQ) API

Required Parameter Group:

1	Qualified user queue name	Input	Char(20)
2	Extended attribute	Input	Char(10)
3	Queue type	Input	Char(1)
4	Key length	Input	Binary(4)
5	Maximum message size	Input	Binary(4)
6	Initial number of messages	Input	Binary(4)
7	Additional number of messages	Input	Binary(4)
8	Public authority	Input	Char(10)
9	Text description	Input	Char(50)

Optional Parameter Group 1:

10	Replace	Input	Char(10)
11	Error code	I/O	Char(*)

Optional Parameter Group 2:

12	Domain	Input	Char(10)
13	Pointers	Input	Char(10)

Optional Parameter Group 3:

14	Number of queue extensions	Input	Binary(4)
15	Reclaim storage	Input	Char(1)

Default Public Authority: *USE

Threadsafe: Yes

The Create User Queue (QUSCRTUQ) API creates a user queue in either the user domain or the system domain, based on the input parameters. A user-domain user queue can be directly manipulated with MI instructions. If you are running at security level 40 or greater, you cannot directly manipulate a system-domain user queue using MI instructions.

If the number of queue extensions is not specified, user queues can grow to a maximum of 16MB in size, though some of that is used for the queue definition. If the number of queue extensions is specified and non-zero, the queue will be extended by the number of additional messages specified until either the number of queue extensions is reached or the storage limit of 32MB is reached.

The system can automatically extend user queues, so you should create a small queue and allow the system to extend the queue as needed. Smaller queues provide better performance and use less storage. However,

the system does not automatically reduce the size of the queue if it is too large. If you specify the additional number of messages as 0, the system does not extend the user queue you create when the queue is full.

To decrease the size of a user queue, a user must delete the queue and create it again.

There are no APIs to manipulate user queue entries. If you must access a user queue object in a library that does not permit user-domain user objects, you must use data queue objects. For information about data queues, refer to the CL Programming book.

If you need to know into which domain the user queue object was created, use either of the following APIs to retrieve that information:

- Retrieve Object Description (QUSROBJD) API
- List Object (QUSLOBJ) API

Note: For performance reasons, the *USRQ object is created before checking to see if it exists in the library specified for the qualified user queue name. If you have an application using this API repeatedly, even if you are using *NO for the replace parameter, permanent system addresses will be used.

Note: User queues created with either the number of queue extensions parameter specified with a positive, non-zero number or the reclaim storage parameter specified to reclaim the user queue storage are not eligible to save to a pre-V4R5 system.

Authorities and Locks

Library Authority

*READ and *ADD.

User Queue Authority

*OBJMGT, *OBJEXIST, and *READ. These authorities are required only if the replace parameter is used and if there is an existing user queue to replace.

User Queue Lock

*EXCL. This applies to both the user queue being created and an existing user queue being replaced.

Required Parameter Group

Qualified user queue name

INPUT; CHAR(20)

The first 10 characters contain the user queue name, and the second 10 characters contain the name of the library where the user queue is located. The only special value supported is *CURLIB.

User queues created in the QTEMP and QRPLOBJ libraries are not forced to permanent storage; they are deleted when those libraries are cleared at sign-off and system IPL, respectively.

Extended attribute

INPUT; CHAR(10)

The extended attribute of the user queue. For example, an object type of *FILE has an extended attribute of PF (physical file), LF (logical file), DSPF (display file), SAVF (save file), and so on.

The extended attribute must be a valid *NAME. You can enter this parameter in uppercase, lowercase, or mixed case. The API automatically converts it to uppercase.

Queue type

INPUT; CHAR(1)

The sequence in which messages are to be dequeued from the queue. The valid values are:

F First-in first-out

K Keyed

L Last-in first-out

Key length

INPUT; BINARY(4)

The length in bytes of the message key from 1 to 256 if you specify the queue type as keyed. If you specify that the queue is not a keyed queue, the value must be 0.

Maximum message size

INPUT; BINARY(4)

The maximum allowed size of messages to be placed on the queue. The API truncates messages that are longer than the value you specify. The maximum size allowed is 64 000 bytes.

Initial number of messages

INPUT; BINARY(4)

The initial number of messages that the queue can contain.

Additional number of messages

INPUT; BINARY(4)

The amount to increase the maximum number of messages value when the queue is full. When this parameter is set to 0, the queue is not extended if an overflow occurs and an error message is sent to the program attempting to place a message on the queue.

Public authority

INPUT; CHAR(10)

The authority you give to the users who do not have specific private or group authority to the user queue. Once the user queue has been created, its public authority stays the same when it is moved to another library or restored from backup media.

If the replace parameter is used and a user queue exists to be replaced, this parameter is ignored. All authorities are transferred from the replaced user queue to the new one.

The valid values for this parameter are:

**ALL*	The user can perform all authorized operations on the user queue.
Authorization list name	The user queue is secured by the specified authorization list, and its public authority is set to *AUTL. The specified authorization list must exist on the system when this API is issued. If it does not exist, the create process fails, and an error message is returned to the application.

**CHANGE*	The user has read, add, update, and delete authority to the user queue and can read the object description.
**EXCLUDE*	The user cannot access the user queue in any way.
**LIBCRTAUT*	The public authority for the user queue is taken from the CRTAUT value for the target library when the object is created. If the CRTAUT value for the library changes later, that change does not affect user queues already created. If the CRTAUT value contains an authorization list name and that authorization list secures an object, do not delete the list. If you do, the next time you call this API with the *LIBCRTAUT parameter, it will fail.
**USE*	The user can read the object description and the user queue's contents but cannot change them.

Text description

INPUT; CHAR(50)

A brief description of the user queue.

Optional Parameter Group 1

Replace

INPUT; CHAR(10)

Whether to replace an existing user queue. Valid values for this parameter are:

**NO* Do not replace an existing user queue of the same name and library. *NO is the default.

**YES* Replace an existing user queue of the same name and library.

The user queue being replaced is destroyed if both:

- The user queue you are replacing is in the user domain.
- The allow user domain (QALWUSRDMN) system value is not set to *ALL or does not contain the library QRPLOBJ.

If the QRPLOBJ library is specified in the QALWUSRDMN system value, then the replaced user-domain user queue is moved to the QRPLOBJ library. If the user queue is in the system domain, it is moved to the QRPLOBJ library, which is cleared at system IPL. For details about authorities, ownership, and renaming, see the discussion of the REPLACE parameter in Control Language (CL) information.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Optional Parameter Group 2

Domain

INPUT; CHAR(10)

The domain into which the user queue should be created. If this parameter is not specified, the value of *DEFAULT will be assumed by the API. Valid values for this parameter are:

**DEFAULT*	Allows the system to decide into which domain the object should be created.
**SYSTEM*	Creates the user queue object into the system domain. The API can always create a user queue into the system domain regardless of the security level you are running at. However, system-domain user queues can only be used at security level 30 and below because there are no APIs to access user queues.
**USER*	Attempts to create the user queue object into the user domain. This is not always possible. If the library you are creating the user queue into does not appear in the QALWUSRDMN system value, the API cannot create the user queue into the user domain. An error message will be returned.

The API uses the following criteria to determine into which domain to create the user queue. The destination library is the library you specified in the qualified user queue name parameter. The optional domain parameter is the information specified in the domain parameter.

QALWUSRDMN System Value	**Destination Library**	**Optional Domain Parameter**	**Domain of Created Object**
*ALL	Any	*DEFAULT	User domain
*ALL	Any	*SYSTEM	System domain
*ALL	Any	*USER	User domain
QTEMP	QTEMP	*DEFAULT	User domain
QTEMP	QTEMP	*SYSTEM	System domain
QTEMP	QTEMP	*USER	User domain
Does not contain library name	Library name	*DEFAULT	System domain
Does not contain library name	Library name	*SYSTEM	System domain
Does not contain library name	Library name	*USER	None; error is returned
Note: The QALWUSRDMN system value lists the libraries into which the user domain objects can be created. The libraries include special value *ALL or a list of one or more library names.			

Pointers

INPUT; CHAR(10)

Whether or not the user queue messages can contain pointer data or not. If this parameter is not specified, *NO is assumed.

*YES Messages can contain pointer and scalar data. Messages to be enqueued must be 16-byte aligned, regardless of whether or not the message contains pointer data. Only user-domain user queues can contain pointer data; therefore, queues that support pointers cannot be created in or restored to a library that is not permitted by system value QALWUSRDMN. User queues that can contain pointers cannot be saved to a release prior to Version 2 Release 3 Modification Level 0.

*NO Messages can contain scalar data only. (User queues created prior to Version 2 Release 3 Modification Level 0 contain scalar data only). The user queue can be in either the system domain or the user domain.

Optional Parameter Group 3

Number of queue extensions

INPUT; BINARY(4)

The maximum number of extensions allowed for the user queue. A value of -1 indicates that the maximum number of extensions will be chosen by the machine. If this parameter is not specified, 0 is assumed.

Reclaim storage

INPUT; CHAR(1)

Whether storage reclaim will be attempted when the number of currently enqueued messages in the user queue reaches zero. Allowable values are:

0 Do not reclaim storage when the user queues have no currently enqueued messages. 0 is the default value.

1 Reclaim storage when the user queues have no currently enqueued messages.

Error Messages

Message ID	Error Message Text
CPF2143 E	Cannot allocate object &1 in &2 type *&3.
CPF2144 E	Not authorized to &1 in &2 type *&3.
CPF2283 E	Authorization list &1 does not exist.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C02 E	User queue &1 not created.
CPD3C01 D	Object name &1 is not valid.
CPD3C03 D	Extended attribute &1 is not valid.

CPD3C05 D Value &1 for authority parameter is not valid.

CPD3C06 D Number of messages requested, &1, is too large for this queue.

CPD3C07 D Value &1 for queue type parameter is not valid.

CPD3C08 D Initial number of queue messages specified, &1, is not valid.

CPD3C09 D Extension parameter value &1 for queue overflow is not valid.

CPD3C10 D Value &1 for key length parameter is not valid.

CPD3C11 D Value &1 for maximum message size parameter is not valid.

CPF3C08 E Initial number of queue messages specified, &1, is not valid.

CPF3C09 E Extension parameter value &1 for queue overflow is not valid.

CPF3C10 E Value &1 for key length parameter is not valid.

CPF3C11 E Value &1 for maximum message size parameter is not valid.

CPF3C2B E Extended attribute &1 is not valid.

CPF3C2D E Value &1 for authority parameter is not valid.

CPF3C2E E Number of messages requested, &1, is too large for this queue.

CPF3C2F E Value &1 for queue type parameter not valid.

CPF3C29 E Object name &1 is not valid.

CPF3C34 E Value &1 for replace option is not valid.

CPF3C36 E Number of parameters, &1, entered for this API was not valid.

CPF3C45 E Value &1 not valid for domain parameter.

CPF3C46 E Value &1 not valid for pointers parameter.

CPF3C47 E Pointers not valid for system domain user queue.

CPF3C49 E Request for user domain object cannot be granted.

CPF3C90 E Literal value cannot be changed.

CPF3C94 E Value &1 not valid for reclaim storage parameter.

CPF8100 E All CPF81xx messages could be returned. xx is from 01 to FF.

CPF9810 E Library &1 not found.

CPF9820 E Not authorized to use library &1.

CPF9830 E Cannot assign library &1.

CPF9838 E User profile storage limit exceeded.

CPF9870 E Object &2 type *&5 already exists in library &3.

CPF9872 E Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V1R3

Top | Object APIs | APIs by category

Delete User Queue (QUSDLTUQ) API

Required Parameter Group:

1	Qualified user queue name	Input	Char(20)
2	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Delete User Queue (QUSDLTUQ) API deletes user queues created with the Create User Queue (QUSCRTUQ) API. The Delete User Queue (DLTUSRQ) command has the same function.

Authorities and Locks

Library Authority

*EXECUTE

User Queue Authority

*OBJEXIST

User Queue Lock

*EXCL

Required Parameter Group

Qualified user queue name

INPUT; CHAR(20)

The name of the user queue and the name of the library in which it resides. The first 10 characters contain the user queue name, and the second 10 characters contain the library name.

The user queue name can be either a specific name or a generic name, a string of one or more characters followed by an asterisk (*). If you specify a generic name, QUSDLTUS deletes all user queues that have names beginning with the string for which the user has authority.

You can use these special values for the library name:

**ALL*	All libraries
**ALLUSR*	All user-defined libraries, plus libraries containing user data and having names starting with Q
**CURLIB*	The job's current library
**LIBL*	The library list

**USRLIBL* The user portion of the job's library 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
CPF2105 E	Object &1 in &2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2113 E	Cannot allocate library &1.
CPF2114 E	Cannot allocate object &1 in &2 type *&3.
CPF2117 E	&4 objects type *&3 deleted. &5 objects not deleted.
CPF2125 E	No objects deleted.
CPF2176 E	Library &1 damaged.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1 in &2 type *&3.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R1

User Index APIs

The user index APIs allow you to:

- Create and delete user indexes
- Add, retrieve, and remove user index entries
- Retrieve the attributes of a user index

For additional information, see Using User Index APIs.

The user index APIs are:

- Add User Index Entries (QUSADDUI) allows you to add one or more new entries or replace existing entries into the user index.
- Create User Index (QUSCRTUI) creates a user index.
- Delete User Index (QUSDLTUI) deletes a user index.
- Remove User Index Entries (QUSRMVUI) removes one or more user index entries that match the remove criteria specified.
- Retrieve User Index Attributes (QUSRUIAT) retrieves information about the user index attributes, including when it was created. It also retrieves the current operational statistics of the user index.
- Retrieve User Index Entries (QUSRTVUI) retrieves user index entries that match the search criteria specified.

Using User Index APIs

A **user index** is an object that allows search functions for data in the index and automatically sorts data based on the value of the data. User indexes are permanent objects in the user domain or in the system domain. They have an object type of *USRIDX and a maximum size of »1 terabyte (1 099 511 627 776 bytes).« They help streamline table searching, cross-referencing, and ordering of data. In general, if your table is longer than 1000 entries, an index performs faster than a user-sorted table.

You can use user indexes to:

- Provide search functions
- Do faster insert operations than in a database file
- Do faster retrieve operations than in a database file
- Create an index by name, such as a telephone directory
- Use order entry programs
- Look up abbreviations in an index
- Sort data automatically based on the hexadecimal value of a key

For more information about user index considerations, refer to the System API Programming book on the V5R1 Supplemental Manuals Web site. User index entries cannot contain a pointer. You can save and restore all the data in an index. You can also save and restore user indexes to another system.

In addition to the user index APIs, you can work with user indexes through the following:

- ILE C programming language
- Machine interface (MI) instructions
- Delete User Index (DLTUSRIDX) command

For details about MI instructions, refer to the iSeries Machine Interface Instructions, which provides detailed descriptions of the iSeries machine interface instruction fields and the formats of those fields. For details about the DLTUSRIDX command, see the Control Language (CL) topic.

Add User Index Entries (QUSADDUI) API

Required Parameter Group:

1	Returned library name	Output	Char(10)
2	Number of entries added	Output	Binary(4)
3	Qualified user index name	Input	Char(20)
4	Insert type	Input	Binary(4)
5	Index entries	Input	Char(*)
6	Length of index entries	Input	Binary(4)
7	Entry lengths and offsets	Input	Array(*) of Char(8)
8	Number of entries	Input	Binary(4)
9	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Add User Index Entries (QUSADDUI) API inserts one or more entries into the user index by the insert type. Each entry is inserted into the index at the appropriate location based on the binary value of the entry. No other collating sequence is supported. Every entry added causes the number of entries added parameter to be incremented by 1; you can retrieve the current number of entries added by using the Retrieve User Index Attributes (QUSRUIAT) API.

When you request to add multiple entries to a user index, the request may be partially successful in the following situations:

- One or more of the entries you requested to add already exists in the index.
- The index becomes full as the entries are added.
- The user profile storage limit is exceeded as entries are added.
- One or more of the entry lengths or offsets are not valid.

When an error occurs, you should check the number of entries added parameter to see if all entries were successfully added. If the number of entries added parameter and the number of entries parameter are not equal, then all entries were not added.

Note: If you add new entries with an entry length longer than the user index is expecting, the entries are truncated to the right. No error is given.

If you are using a fixed length index and the entries you are adding are less than the fixed length, you may get undesirable results. The entries are not padded with blanks before being entered into the user index. The API checks the length of index entries parameter to ensure that the length you pass is a multiple of the length of one index entry.

Authorities and Locks

User Index Library Authority

*EXECUTE

User Index Authority

*CHANGE

User Index Lock

*SHRUPD

Required Parameter Group

Returned library name

OUTPUT; CHAR(10)

The name of the library that contains the user index to which the entries were added if they were added successfully. This parameter is not set if no entries were successfully added. This information helps you identify the specific library used when *LIBL or *CURLIB is specified in the qualified user index name parameter.

Number of entries added

OUTPUT; BINARY(4)

The number of entries successfully added to the specified user index. If an error is received while processing the entries, this number indicates how many were added before the error occurred.

Qualified user index name

INPUT; CHAR(20)

The user index to which you want to add entries and the library in which it is located. The first 10 characters contain the user index name, and the second 10 characters contain the library name.

You can use these special values for the library name:

**CURLIB*	The job's current library
**LIBL*	The library list

Insert type

INPUT; BINARY(4)

The type of insert to be performed against all entries that are to be added.

Valid values are:

1	Insert unique argument This value is only valid for nonkeyed user indexes. If the entry is already in the index or the index entries are keyed, an error is returned.

2 Insert with replacement

This value requests to replace the nonkey portion of the index entry if the key is already in the user index. It is only valid for keyed user indexes. If the entry does not exist, it will be inserted into the user index. If the index entries are nonkeyed, an error is returned.

3 Insert without replacement

This value requests to insert the entry only if the key is not already in the user index. It is only valid for a keyed user index. If the entry does not exist, it will be inserted into the user index. If the entry is already in the index or the index entries are nonkeyed, an error is returned.

Index entries

INPUT; CHAR(*)

The actual entry or entries to be added to the user index. If the user index contains *fixed length* entries, this parameter is processed using the entry length specified when the user index was created. If the user index contains *variable length* entries, this parameter is processed using the information contained in the entry lengths and entry offsets parameter.

When using an index that contains fixed length entries, this parameter should be the same length as the length of index entries parameter.

Length of index entries

INPUT; BINARY(4)

The length of the index entries parameter. This value must be greater than 0.

Entry lengths and entry offsets

INPUT; ARRAY(*) of CHAR(8)

If the user index contains variable length entries, this parameter is a data-structure array that is used to parse through the index entries parameter. In this case, an entry length and entry offset need to be provided for all entries that are to be added. This parameter is ignored for user indexes containing fixed length entries.

The size of the entry lengths and entry offsets parameter must be at least eight times the number of entries parameter; otherwise, an error will be returned.

See Format for Entry Lengths and Entry Offsets for details on the data structure.

Number of entries

INPUT; BINARY(4)

The number of entries that are to be added to the user index. Valid values are 1 through 4095.

Error code

I/O; CHAR(*)

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

Format for Entry Lengths and Entry Offsets

The following table defines the format for the entry lengths and entry offsets parameter. This information is needed to parse through the index entries parameter.For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
Note: The following fields will be repeated. The number of times they are repeated depends on the value specified in the number of entries parameter.			
		BINARY(4)	Entry length
		BINARY(4)	Entry offset

Field Descriptions

Entry length The length of the entry to be inserted into the index. Valid values are 1-2000. This value depends on how the user index was created.

Entry offset. For the first entry, the offset is the number of bytes from the beginning of the index entries parameter to the first byte of the first entry. For each subsequent entry, the offset is the number of bytes from the beginning of the previous entry to the first byte of the next entry. Each entry offset value must be greater than or equal to 0 and must refer to an entry within the index entries parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C7B E	Length of index entries is not valid.
CPF3C7C E	User index is full.
CPF3C71 E	Insert type &1 is not valid.
CPF3C72 E	Number of entries &1 is not valid.
CPF3C73 E	Error occurred with index entries parameter.
CPF3C74 E	Entry is already in the index.
CPF3C75 E	Error occurred with entry lengths and offsets parameter.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
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.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Top | Object API categories | API by category

Create User Index (QUSCRTUI) API

Required Parameter Group:

1	Qualified user index name	Input	Char(20)
2	Extended attribute	Input	Char(10)
3	Entry length attribute	Input	Char(1)
4	Entry length	Input	Binary(4)
5	Key insertion	Input	Char(1)
6	Key length	Input	Binary(4)
7	Immediate update	Input	Char(1)
8	Optimization	Input	Char(1)
9	Public authority	Input	Char(10)
10	Text description	Input	Char(50)

Optional Parameter Group 1:

11	Replace	Input	Char(10)
12	Error code	I/O	Char(*)

Optional Parameter Group 2:

13	Domain	Input	Char(10)

Optional Parameter Group 3:

14	Usage tracking	Input	Char(1)

»Optional Parameter Group 4:

15	Index size option	Input	Char(1)«

Default Public Authority: *USE

Threadsafe: Yes

The Create User Index (QUSCRTUI) API creates a user index in either the user domain or the system domain. A system-domain user index cannot be saved to a release prior to Version 2 Release 3 Modification 0. A user-domain user index can be directly manipulated with MI instructions and can also be accessed using system APIs at all security levels. On a system with the QSECURITY system value set to 40 or greater, you must use system APIs to access system-domain user indexes. If you create a permanent object by using this API, you cannot delete the object by using the MI instruction DESINX when the system security level is set to 40 or greater. You would have to delete the object by using the Delete User Index (QUSDLTUI) API.

»**Note:** If the user index is larger than 4 gigabytes, it cannot be saved to a release prior to Version 5 Release 2 Modification 0.«

If the user index was created prior to Version 2 Release 2 Modification 0, the size of the user index is limited to a maximum of 1 gigabyte. (A user index with a size greater than 1 gigabyte cannot be saved to or restored from a release prior to Version 2 Release 2 Modification 0.) If the user index was created on or after Version 2 Release 2 Modification 0, the size of the object is limited to a maximum of 4 gigabytes. The size is dependent on the amount of storage needed for the number and size of index entries and excludes the size of the associated space, if any.

Note: You can tell whether a user index object can be saved to a release prior to Version 2 Release 2 Modification 0:

- By ensuring that the current object size is less than 1 gigabyte by using one of the following:
  - The Display Object Description (DSPOBJD) command
  - The List Objects (QUSLOBJ) API
  - The Retrieve Object Description (QUSROBJD) API
- By ensuring that the key length field is 120 bytes or less by using either of the following:
  - The Materialize Index Attributes (MATINXAT) MI instruction
  - The Retrieve User Index Attributes (QUSRUIAT) API

Note: For performance reasons, the *USRIDX object is created before checking to see if it exists in the library specified for the qualified user index name. If you have an application using this API repeatedly, even if you are using *NO for the replace parameter, permanent system addresses will be used.

Authorities and Locks

Library Authority

*READ and *ADD.

User Index Authority

*OBJMGT, *OBJEXIST, and *READ. These authorities are required only if the replace parameter is used and there is an existing user index to replace.

User Index Lock

*EXCL. This applies to both the user index being created and an existing user index being replaced.

Required Parameter Group

Qualified user index name

INPUT; CHAR(20)

The name of the user index being created, and the library in which it is to be located. The first 10 characters contain the user index name, and the second 10 characters contain the library name.

You can use this special value for the library name:

**CURLIB* The job's current library

User indexes created in the QTEMP and QRPLOBJ libraries are not forced to permanent storage; they are deleted when those libraries are cleared at sign-off and system IPL, respectively.

Extended attribute

INPUT; CHAR(10)

The extended attribute of the user index. For example, an object type of *FILE could have an extended attribute of PF (physical file), LF (logical file), DSPF (display file), or SAVF (save file).

The extended attribute must be a valid *NAME. You can enter this parameter in uppercase, lowercase, or mixed case. The API automatically converts it to uppercase.

Entry length attribute

INPUT; CHAR(1)

Whether there are fixed-length or variable-length entries in the user index. The valid values are:

F Fixed-length entries

V Variable-length entries

Entry length

INPUT; BINARY(4)

The length of entries in the index.

The valid values for fixed-length entries are from 1 through 2000.

Valid values for variable length entries are 0 or -1. A value of 0 enables a maximum entry length of 120 bytes and a key length from 1 through 120. A value of -1 enables a maximum entry length of 2000 and a key length from 1 through 2000.

Note: A user index created with an entry length greater than 120 cannot be saved or restored to a release prior to Version 2 Release 2 Modification 0.

Key insertion

INPUT; CHAR(1)

Whether the inserts to the index are by key. The valid values are:

0 No insertion by key

1 Insertion by key

Key length

INPUT; BINARY(4)

The length of the key where the first byte of an entry is the beginning of the key for the index entries. The value for this parameter must be 0 for no insertion by key. If you specify key length insertion, this value is from 1 through 2000.

Immediate update

INPUT; CHAR(1)

Whether the updates to the index are written synchronously to auxiliary storage on each update to the index. The valid values are:

0 No immediate update

1 Immediate update

Each update to the index is written to auxiliary storage after every insert and remove operation.

Optimization

INPUT; CHAR(1)

The type of access in which to optimize the index. The valid values are:

0 Optimize for random references

1 Optimize for sequential references

Public authority

INPUT; CHAR(10)

The authority you give to users who do not have specific private or group authority to the user index. Once the user index has been created, its public authority stays the same when it is moved to another library or restored from backup media.

If the replace parameter is used and an existing user index is replaced, this parameter is ignored. All authorities are transferred from the replaced user index to the new one.

The valid values for this parameter are:

**ALL*	The user can perform all authorized operations on the user index.
Authorization list name	The user index is secured by the specified authorization list, and its public authority is set to *AUTL. The specified authorization list must exist on the system when this API is issued. If the list does not exist, the create process fails, and an error message is returned to the application.
**CHANGE*	The user has read, add, update, and delete authority for the user index and can read the object description.
**EXCLUDE*	The user cannot access the user index in any way.
**LIBCRTAUT*	The public authority for the user index is taken from the CRTAUT value for the target library when the object is created. If the CRTAUT value for the library changes later, that change does not affect user indexes already created. If the CRTAUT value contains an authorization list name and that authorization list secures an object, do not delete the list. If you do, the next time you call this API with the *LIBCRTAUT parameter, it will fail.
**USE*	The user can read the object description and contents but cannot change the user index.

Text description

INPUT; CHAR(50)

A brief description of the user index.

Optional Parameter Group 1

Replace

INPUT; CHAR(10)

Whether you want to replace an existing user index. Valid values for this parameter are:

**NO* Do not replace an existing user index of the same name and library. *NO is the default value.

**YES* Replace an existing user index of the same name and library.

If the user index already exists, you can replace it with a new user index of the same name and library. The new user index is subject to the same authorities. The user index being replaced is destroyed if both:

- The allow user domain (QALWUSRDMN) system value is not set to *ALL or does not contain the QRPLOBJ library.
- The user index you are replacing is in the user domain.

If the user index is in the system domain, it is moved to the QRPLOBJ library. If QALWUSRDMN is set to *ALL or if it contains QRPLOBJ, the replaced user index is moved to QRPLOBJ, which is cleared at system IPL. For details about authorities, ownership, and renaming, see the discussion of the REPLACE parameter in the Control Language (CL) information.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Optional Parameter Group 2

Domain

INPUT; CHAR(10)

The domain into which the user index should be created. If this parameter is not specified, the value of *DEFAULT will be assumed by the API. Valid values for this parameter are:

**DEFAULT* Allows the system to decide into which domain the object should be created.

**SYSTEM* Creates the user index object into the system domain. The API can always create a user index into the system domain, regardless of the security level running. However, if you are running at security level 40 or greater, you must use APIs to access system-domain user index objects.

**USER* Attempts to create the user index object into the user domain. This is not always possible. If the library you are creating the user index into does not appear in the QALWUSRDMN system value, the API cannot create the user index into the user domain. An error message will be returned.

The API uses the following criteria to determine into which domain to create the user index. The destination library is the library you specified in the qualified user index name parameter. The optional domain parameter is the information specified in the domain parameter.

QALWUSRDMN System Value	**Destination Library**	**Optional Domain Parameter**	**Domain of Created Object**
*ALL	Any	*DEFAULT	User domain
*ALL	Any	*SYSTEM	System domain
*ALL	Any	*USER	User domain
QTEMP	QTEMP	*DEFAULT	User domain
QTEMP	QTEMP	*SYSTEM	System domain
QTEMP	QTEMP	*USER	User domain
Does not contain library name	Library name	*DEFAULT	System domain
Does not contain library name	Library name	*SYSTEM	System domain
Does not contain library name	Library name	*USER	None; error is returned
Note: The QALWUSRDMN system value lists the libraries into which the user domain objects can be created. Valid libraries are the special value *ALL or a list of one or more library names.			

If your system is at security level 40 or greater, you must use APIs to access system-domain user indexes using APIs. You cannot use MI instructions to directly access system-domain user indexes.

You can use the Retrieve Object Description (QUSROBJD) API or the List Object (QUSLOBJ) API to determine into which domain the user index object was created.

Optional Parameter Group 3

Usage tracking

INPUT; CHAR(1)

The usage tracking state. Usage tracking provides machine checkpoints to improve availability of user indexes. If a user index is found to be a state of partial change, it will be marked as damaged. The valid values are:

0 Do not track usage state. 0 is the default value.

1 Track usage state.

»Optional Parameter Group 4

Index size option

INPUT; CHAR(1)

The maximum size of the user index. The valid values are:

0 The maximum size of the user index is 4 gigabytes.

1 The maximum size of the user index is 1 terabyte.«

Dependencies between Parameters

Some of the parameters are interdependent and are shown in the following table:

Entry Length Attribute	Entry Length (n)	Key Insertion	Key Length (x)
Fixed	1 <= n <= 2000	No	0
Fixed	1 <= n <= 2000	Yes	1 <= x <= n
Note: For the following entry lengths: ● 0 signifies a maximum entry length of 120. ● -1 signifies a maximum entry length of 2000.			
Variable	0, -1	No	0
Variable	0	Yes	1 <= x <= 120
Variable	-1	Yes	1 <= x <= 2000

Error Messages

Message ID	Error Message Text
CPF2143 E	Cannot allocate object &1 in &2 type *&3.
CPF2144 E	Not authorized to &1 in &2 type *&3.
CPF2283 E	Authorization list &1 does not exist.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C0A E	Value &1 for entry length parameter is not valid.
CPF3C0B E	Value &1 for immediate update parameter is not valid.
CPF3C0C E	Value &1 for key length parameter is not valid.

CPF3C0D E Value &1 for key insertion parameter is not valid.

CPF3C0E E Value &1 for the optimization parameter is not valid.

CPF3C03 E User index &2 not created.

CPD3C01 D Object name &1 is not valid.

CPD3C02 D Value &1 for entry length attribute parameter is not valid.

CPD3C03 D Extended attribute &1 is not valid.

CPD3C05 D Value &1 for authority parameter is not valid.

CPD3C0A D Value &1 for entry length parameter is not valid.

CPD3C0B D Value &1 for immediate update parameter is not valid.

CPD3C0C D Value &1 for key length parameter is not valid.

CPD3C0D D Value &1 for key insertion parameter is not valid.

CPD3C0E D Value &1 for the optimization parameter is not valid.

CPF3C2A E Value &1 for entry length attribute parameter is not valid.

CPF3C2B E Extended attribute &1 is not valid.

CPF3C2D E Value &1 for authority parameter is not valid.

CPF3C29 E Object name &1 is not valid.

CPF3C34 E Value &1 for replace option is not valid.

CPF3C36 E Number of parameters, &1, entered for this API was not valid.

CPF3C45 E Value &1 not valid for domain parameter.

CPF3C49 E Request for user domain object cannot be granted.

CPF3C90 E Literal value cannot be changed.

CPF3C93 E Value &1 not valid for usage tracking parameter.

»CPF3C95 E Value &1 not valid for index size option parameter.«

CPF8100 E All CPF81xx messages could be returned. xx is from 01 to FF.

CPF9810 E Library &1 not found.

CPF9820 E Not authorized to use library &1.

CPF9830 E Cannot assign library &1.

CPF9838 E User profile storage limit exceeded.

CPF9870 E Object &2 type *&5 already exists in library &3.

CPF9872 E Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V1R3

Delete User Index (QUSDLTUI) API

Required Parameter Group:

1	Qualified user index name	Input	Char(20)
2	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Delete User Index (QUSDLTUI) API deletes user indexes created with the Create User Index (QUSCRTUI) API.

The QUSDLTUI API performs the same function as the Delete User Index (DLTUSRIDX) command.

Authorities and Locks

Library Authority

*EXECUTE

User Index Authority

*OBJEXIST

User Index Lock

*EXCL

Required Parameter Group

Qualified user index name

INPUT; CHAR(20)

The name of the user index and the name of the library in which it resides. The first 10 characters contain the user index name, and the second 10 characters contain the library name. The user index name can be either a specific name or a generic name, a string of one or more characters followed by an asterisk (*). If you specify a generic name, QUSDLTUI deletes all user indexes that have names beginning with the string for which the user has authority.

You can use these special values for the library name:

**ALL*	All libraries
**ALLUSR*	All user-defined libraries, plus libraries containing user data and having names starting with Q. For information on the libraries included, see *ALLUSR.
**CURLIB*	The job's current library

**LIBL* The library list

**USRLIBL* The user portion of the job's library 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
CPF2105 E	Object &1 in &2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2113 E	Cannot allocate library &1.
CPF2114 E	Cannot allocate object &1 in &2 type *&3.
CPF2117 E	&4 objects type *&3 deleted. &5 objects not deleted.
CPF2125 E	No objects deleted.
CPF2176 E	Library &1 damaged.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1 in &2 type *&3.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R1

Remove User Index Entries (QUSRMVUI) API

Required Parameter Group:

1	Number of entries removed	Output	Binary(4)
2	Entries removed	Output	Char(*)
3	Length of entries removed	Input	Binary(4)
4	Entry lengths and entry offsets	Output	Array(*) of Char(8)
5	Length of entry lengths and offsets	Input	Binary(4)
6	Returned library name	Output	Char(10)
7	Qualified user index name	Input	Char(20)
8	Format	Input	Char(8)
9	Maximum number of entries	Input	Binary(4)
10	Remove type	Input	Binary(4)
11	Remove criteria	Input	Char(*)
12	Length of remove criteria	Input	Binary(4)
13	Remove criteria offset	Input	Binary(4)
14	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Remove User Index Entries (QUSRMVUI) API removes one or more user index entries that match the values specified on the remove criteria parameter. It returns the number of entries that were removed and, optionally, returns the actual index entries removed.

Authorities and Locks

User Index Library Authority

*EXECUTE

User Index Authority

*CHANGE

User Index Lock

*SHRUPD

Required Parameter Group

Number of entries removed

OUTPUT; BINARY(4)

The number of index entries, satisfying the values specified on the remove criteria parameter, that

were successfully removed from the user index. If this field is 0, no entries satisfied the remove criteria. This value can never be greater than the maximum number of entries parameter.

Entries removed

OUTPUT; CHAR(*)

The actual entries removed. All entries that satisfied the remove criteria parameter and were removed (up to the maximum number of entries parameter) are returned if sufficient space is provided. The API returns only the data that the area can hold.

The size of the entries removed parameter should be greater than or equal to:

```
8 + (the maximum number of entries parameter
      * the maximum entry length)
```

The maximum entry length was defined when the index was created. It can be obtained by using the Retrieve User Index Attributes (QUSRUIAT) API.

To determine if all the entries are valid in the entries removed parameter, compare the bytes returned and the bytes available fields in the entries removed parameter.

The entries are always returned starting with the entry that is closest to or equal to the remove argument. Then entries are kept in the order that they proceed away from the remove criteria parameter. Each entry removed from the user index is based on the binary value of the remove criteria. No other collating sequence is supported. User indexes can contain only scalar data, which makes the index entries contiguous. Use the entry lengths and entry offsets parameter to parse the entries that were removed and returned in this parameter.

If you do not want the entries that were removed to be returned in this parameter, specify 0 for the length of entries removed parameter.

Every entry removed causes the number of entries removed parameter to be incremented by 1. You can also use the Retrieve User Index Attributes (QUSRUIAT) API to retrieve this information.

Refer to IDXE0100 Format for the layout of this parameter.

Length of entries removed

INPUT; BINARY(4)

The length of the entries removed parameter. If this length is larger than the actual size of the entries removed parameter, the results may not be predictable. The minimum length is 0 or >= (greater than or equal to) 8 bytes. If 0 is used, the entries removed from the index are not returned and the bytes returned and the bytes available in the entries removed parameter are not set.

Entry lengths and entry offsets

OUTPUT; ARRAY of CHAR(8)

A data structure that contains entry lengths and entry offsets for all entries that were found that met the remove criteria parameter. An entry length and entry offset exist for every entry returned in the entries removed parameter. These entry lengths and entry offsets are used to parse through the entries removed parameter. If the length of entries removed parameter is 0, this information will not be returned.

The size of the entry lengths and entry offsets parameter should be at least:

```
8 + (the maximum number of entries parameter * 8)
```

You must provide enough space in both the entries removed and the entry lengths and offset parameter for this API to return complete information to you.

You will not receive complete information in the following two situations.

- You provide enough space in the entries removed parameter for the API to return all index entries removed, but there is not enough space in the entry lengths and entry offset parameter to return the lengths and offsets for all entries. You will be unable to parse through all of the entries in the entries removed parameter. Check the bytes returned and bytes available fields in the entry lengths and entry offsets parameter to ensure the information is complete.
- You provide enough space in the entry lengths and entry offsets parameter to return all lengths and offsets, but there is not enough space in the entries removed parameter to return all index entries removed. Some of the entry lengths and entry offsets will not be valid; they will refer to index entries that could not be returned to you. Check the bytes returned and bytes available fields in the entries removed parameter to ensure that the information is complete.

See Format for Entry Lengths and Entry Offsets for details on the data structure.

Length of entry lengths and entry offsets

INPUT; BINARY(4)

The length of the entry lengths and entry offsets. If the length is longer than the entry lengths and entry offsets parameter, the results may not be predictable. The minimum length is 8. If the length of entries removed parameter is 0, which means you do not want the entries removed to be returned, this parameter is ignored.

Returned library name

OUTPUT; CHAR(10)

The name of the library that contains the user index from which the entries were removed. If the entries are successfully removed from the user index, the name of the library that contained the user index entries is returned. This parameter is not set if an error occurs.

Qualified user index name

INPUT; CHAR(20)

The user index from which you want to remove entries, and the library in which it is located. The first 10 characters contain the user index name, and the second 10 characters contain the library name.

You can use these special values for the library name:

**CURLIB*	The job's current library
**LIBL*	The library list

Format

INPUT; CHAR(8)

The format of the user index entries that were removed.

The format name supported is:

IDXE0100	Basic Information

Refer to IDXE0100 Format for details on the format.

Maximum number of entries

INPUT; BINARY(4)

The maximum number of user index entries to be removed that satisfy the remove criteria. Valid values are 1 through 4095.

Remove type

INPUT; BINARY(4)

The type of remove operation that is to be performed.

Valid values are:

1 Equal

Remove entries that are equal to the remove criteria.

2 Greater than

Remove entries that are greater than the remove criteria.

3 Less than

Remove entries that are less than the remove criteria.

4 Greater than or equal

Remove entries that are greater than or equal to the remove criteria.

5 Less than or equal

Remove entries that are less than or equal to the remove criteria.

6 First

Remove the first index entry or entries.

7 Last

Remove the last index entry or entries.

8 Between

Remove all entries between the two arguments specified in the remove criteria.

Remove criteria

INPUT; CHAR(*)

The criteria used to find matches in the user index.

When the remove type is 8 (between), this parameter contains two criteria elements of the same length. The first element is considered the starting element, and the second element is the ending element. This parameter is ignored when the remove type is 6 (first) or 7 (last).

Length of remove criteria

INPUT; BINARY(4)

The length of the remove criteria being used. This parameter is ignored when the remove type is 6 (first) or 7 (last). If the remove type is 8 (between), this parameter specifies the length of the first element. The second element must have the same length as the first element. Valid values are 1-2000, depending on how the user index was created.

For a fixed and keyed user index, the length of the remove criteria can be greater than the length of the key.

Remove criteria offset

INPUT; BINARY(4)

The offset of the second element from the beginning of the remove criteria parameter. This parameter is ignored unless the remove type is 8 (between).

Error code

I/O; CHAR(*)

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

Format for Entry Lengths and Entry Offsets

The following information is returned in the entry lengths and entry offsets parameter. This information is needed to parse through the entries removed parameter. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
Note: The following fields will be repeated. The number of times they are repeated depends on the length of the entry lengths and entry offsets parameter and the number of entries actually removed.			
		BINARY(4)	Entry length
		BINARY(4)	Entry offset

IDXE0100 Format

The following information is returned for the IDXE0100 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available

8	8	CHAR(*)	Entry 1-n

Field Descriptions

Bytes available. The length of all data available to return. All available data is returned if enough space is provided.

Bytes returned. The length of the data actually returned.

Entry length. The length of the entry removed from the user index. Valid values are 1-2000, depending on how the user index was created.

Entry offset. The number of bytes from the beginning of the immediately preceding entry to the first byte of the entry returned. For the first entry, the offset is the number of bytes from the beginning of the parameter to the first byte of the first entry.

Entry 1-n. All entries that satisfy the remove criteria (up through the maximum number of entries) are returned. User indexes contain only scalar data, which makes the index entries contiguous. Use the entry length and entry offset values to parse this parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C21 E	Format name &1 is not valid.
CPF3C7D E	Remove or search information is not valid.
CPF3C70 E	Length of entries removed parameter is not valid.
CPF3C76 E	Length of lengths and offsets of entries &1 is not valid.
CPF3C77 E	Remove type &1 is not valid.
CPF3C78 E	Criteria length &1 is not valid.
CPF3C79 E	Maximum number of entries &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
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.

CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Top | Object API categories | API by category

Retrieve User Index Attributes (QUSRUIAT) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Qualified user index name	Input	Char(20)
5	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve User Index Attributes (QUSRUIAT) API retrieves information about the current attributes and the current operational statistics of the user index.

Authorities and Locks

User Index Library Authority

*EXECUTE

User Index Authority

*USE

User Index Lock

*SHRUPD

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable that is to receive the information requested. You can specify the size of this area to be smaller than the format requested if you specify the length of receiver variable 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 may not be predictable. The minimum length is 8 bytes.

Format name

INPUT; CHAR(8)

The format of the index information returned.

The format name supported is:

IDXA0100 Basic information

Refer to IDXA0100 Format for details on the format.

Qualified user index name

INPUT; CHAR(20)

The user index for which you want to retrieve information, and the library in which it is located. The first 10 characters contain the user index name, and the second 10 characters contain the library name.

You can use these special values for the library name:

**CURLIB* The job's current library

**LIBL* The library list

Error code

I/O; CHAR(*)

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

IDXA0100 Format

The following information is returned for the IDXA0100 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	User index name
18	12	CHAR(10)	User index library name
28	1C	CHAR(1)	Entry length attribute
29	1D	CHAR(1)	Immediate update
30	1E	CHAR(1)	Key insertion
31	1F	CHAR(1)	Optimized processing mode
32	20	CHAR(4)	Reserved
36	24	BINARY(4)	Entry length
40	28	BINARY(4)	Maximum entry length
44	2C	BINARY(4)	Key length
48	30	BINARY(4)	Number of entries added
52	34	BINARY(4)	Number of entries removed
56	38	BINARY(4)	Number of retrieve operations

Field Descriptions

Bytes available. The length of all data available to return. All available data is returned if enough space is provided.

Bytes returned. The length of the data actually returned.

Entry length. For user indexes with fixed-length entries, this is the length of each index entry. For user indexes with variable-length entries, this is equal to the longest entry that has ever been inserted into the index. Valid values are from 1 through 2000.

Entry length attribute. The types of entries in the user index.

Possible values are:

F Fixed-length entries

V Variable-length entries

Immediate update. Whether or not the updates to the index are written synchronously to auxiliary storage on each update to the index.

The possible values are:

0 No immediate update

1 Immediate update

Key insertion. Whether or not the inserts to the index are by key.

0 No insertion by key

1 Insertion by key

Key length. The length of the key where the first byte of an entry is the beginning of the key for the index entries. This field will be 0 for a nonkeyed user index.

Maximum entry length. The maximum entry length any user index entry can have.

Number of entries added. The number of entries added to the user index. The number of entries currently in the index can be obtained by subtracting the number of entries removed from the number of entries added.

Number of entries removed. The number of entries removed from the user index.

Number of retrieve operations. The number of times either the FNDINXEN (find independent index entry) MI instruction or Retrieve User Index Entry (QUSRTVUI) API has been used on this user index. The QUSRUIAT API or MATINXAT (materialize independent index attributes) MI instruction sets the number of retrieve operations to 0 after the retrieve or materialize operation is completed.

Optimized processing mode. Whether the user index is maintained in a manner that optimizes performance for:

0 Random references

1 Sequential references

Reserved. An ignored field.

User index library name. The name of the library containing the user index. This information is helpful when *CURLIB or *LIBL is specified in the qualified user index name parameter.

User index name. The name of the user index.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
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.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Top | Object API categories | API by category

Retrieve User Index Entries (QUSRTVUI) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Entry lengths and entry offsets	Output	Array(*) of Char(8)
4	Length of entry lengths and offsets	Input	Binary(4)
5	Number of entries returned	Output	Binary(4)
6	Returned library name	Output	Char(10)
7	Qualified user index name	Input	Char(20)
8	Format	Input	Char(8)
9	Maximum number of entries	Input	Binary(4)
10	Search type	Input	Binary(4)
11	Search criteria	Input	Char(*)
12	Length of search criteria	Input	Binary(4)
13	Search criteria offset	Input	Binary(4)
14	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve User Index Entries (QUSRTVUI) API retrieves user index entries that match the criteria specified on the search criteria parameter.

The entries are always returned starting with the entry that is closest to or equal to the search criteria parameter and then proceeding away from the search criteria. The number of entries returned parameter will never exceed the value specified in the maximum number of entries parameter. Each entry retrieved from the user index is based on the binary value of the search criteria parameter. No other collating sequence is supported.

Every entry retrieved causes the number of retrieve operations to be incremented by 1.

Authorities and Locks

User Index Library Authority

*EXECUTE

User Index Authority

*USE

User Index Lock

*SHRUPD

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable that is to receive the information requested. You can specify the size of this 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.

Use the entry lengths and entry offsets parameter to parse through this parameter. If the number of entries returned parameter is 0, then only the bytes available and the bytes provided have been changed.

To determine if all the entries are valid in the receiver variable, compare the bytes returned and bytes available fields. If the bytes returned are less than the bytes available, your receiver variable is not large enough to hold all the entries that match the search criteria parameter. While processing the entries, you need to make sure that both:

- Your current offset in the receiver variable plus the entry offset is less than the length of the receiver variable
- Your current offset in the receiver variable plus the entry offset plus the entry length is less than the length of the receiver variable.

The size of the receiver variable parameter should be greater than or equal to:

```
8 + (the maximum number of entries parameter
      * the maximum entry length)
```

The maximum entry length was defined when the index was created. It can be obtained by using the Retrieve User Index Attributes (QUSRUIAT) API.

Refer to the IDXE0100 Format for the layout of this parameter.

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 may not be predictable. The minimum length is 8 bytes.

Entry lengths and entry offsets

OUTPUT; ARRAY(*) of CHAR(8)

A data structure containing entry lengths and entry offsets for all entries found that met the search criteria. An entry length and entry offset exist for every entry returned in the receiver variable. These entry lengths and entry offsets are used to parse through the receiver variable.

The size of the entry lengths and entry offsets parameter should be at least:

```
8 + (the maximum number of entries parameter * 8)
```

You must provide enough space in both the receiver variable and the entry lengths and entry offsets parameter for this API to return this information to you. You will not receive complete information in the following two situations.

- You provide enough space in the receiver variable parameter for the API to return all index entries, but there is not enough space in the entry lengths and entry offset parameter to return the lengths and offsets for all entries. You will be unable to parse through all of the

entries in the receiver variable parameter. Check the bytes returned and bytes available fields in the entry lengths and entry offsets parameter to ensure that the information is complete.

- You provide enough space in the entry lengths and entry offsets parameter to return all lengths and offsets, but there is not enough space in the receiver variable parameter to return all index entries removed. Some of the entry lengths and entry offsets will not be valid; they will refer to index entries that could not be returned to you. Check the bytes returned and bytes available fields in the receiver variable parameter to ensure that the information is complete.

See the Format for Entry Lengths and Entry Offsets for details about the data structure.

Length of entry lengths and entry offsets

INPUT; BINARY(4)

The length of the entry lengths and entry offsets parameter. If the length is longer than the entry lengths and entry offsets parameter, the results may not be predictable. The minimum length is 8.

If the receiver variable cannot hold all the entries that satisfy the search criteria:

- The entries are truncated.
- Not all the information in the entry lengths and entry offsets parameter is valid.

Number of entries returned

OUTPUT; BINARY(4)

The total number of index entries found that satisfy the search criteria. If this field is 0, no entries satisfied the search criteria. This value can never be greater than the maximum number of entries parameter.

Returned library name

OUTPUT; CHAR(10)

The name of the library that contains the user index from which the entries were successfully retrieved. This parameter is not set if an error occurs.

Qualified user index name

INPUT; CHAR(20)

The user index for which you want to retrieve information, and the library in which it is located. The first 10 characters contain the user index name, and the second 10 characters contain the library name.

You can use these special values for the library name:

**CURLIB* The job's current library

**LIBL* The library list

Format

INPUT; CHAR(8)

The format of the receiver variable.

The format name supported is:

IDXE0100 Basic Information

Refer to IDXE0100 Format for details about the format.

Maximum number of entries

INPUT; BINARY(4)

The maximum number of index entries to be returned that match the search criteria. Valid values are 1 through 4095.

Search type

INPUT; BINARY(4)

The type of search that is to be performed.

Valid values are:

1 Equal

Remove entries that are equal to the remove criteria.

2 Greater than

Remove entries that are greater than the remove criteria.

3 Less than

Remove entries that are less than the remove criteria.

4 Greater than or equal

Remove entries that are greater than or equal to the remove criteria.

5 Less than or equal

Remove entries that are less than or equal to the remove criteria.

6 First

Remove the first index entry or entries.

7 Last

Remove the last index entry or entries.

8 Between

Remove all entries between the two arguments specified in the remove criteria.

Search criteria

INPUT; CHAR(*)

The criteria used to find matches in the user index.

If the search type is 8 (between), both search elements must have the same length. When the search type is 8 (between), this parameter contains two search elements. The first element is considered the starting element, and the second element is the ending element.

This parameter is ignored when the search type parameter is 6 (first) or 7 (last).

Length of search criteria

INPUT; BINARY(4)

The length of the search criteria that is to be used. This parameter is ignored when the search type is 6 (first) or 7 (last).

If the search type is 8 (between), this parameter specifies the length of the first element. The second element must have the same length as the first element. Valid values are 1-2000, depending on how the user index was created.

For a fixed and keyed user index, the length of the search criteria:

- Can be greater than the length of the key
- Must be less than or equal to the entry length

Search criteria offset

INPUT; BINARY(4)

The offset of the second search element from the beginning of the search criteria parameter. This parameter is ignored unless the search type is 8 (between).

Error code

I/O; CHAR(*)

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

Format for Entry Lengths and Entry Offsets

The following information is returned in the entry lengths and entry offsets parameter. The information is needed to parse through the receiver variable. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
Note: The following fields will be repeated. The number of times they are repeated depends on the length of the entry lengths and entry offsets parameter and the number of entries actually retrieved.			
		BINARY(4)	Entry length
		BINARY(4)	Entry offset

IDXE0100 Format

The following index information is returned for the IDXE0100 format in the receiver variable parameter. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(*)	Entry 1-n

Field Descriptions

Bytes available. The length of all data available to return. All available data is returned if enough space is provided.

Bytes returned. The length of the data actually returned.

Entry length. The length of the entry retrieved from the index. Valid values are 1-2000, depending on how the user index was created.

Entry offset. The number of bytes from the beginning of the immediately preceding entry to the first byte of the entry returned. For the first entry, the offset is the number of bytes from the beginning of the receiver variable to the first byte of the first entry.

Entry 1-n. All entries that satisfy the search criteria (up to the maximum number of entries parameter) are returned. User indexes are created to contain only scalar data, which results in the index entries being contiguous. Use the entry length and entry offset values to parse this field.

This field is repeated by the value in the number of entries returned parameter if the receiver variable is large enough to hold all of the entries found.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C7A E	Search type &1 is not valid.

CPF3C7D E	Remove or search information is not valid.
CPF3C76 E	Length of lengths and offsets of entries &1 is not valid.
CPF3C78 E	Criteria length &1 is not valid.
CPF3C79 E	Maximum number of entries &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
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.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Top | Object API categories | API by category

User Space APIs

The user space APIs allow you to create and delete user spaces, change and retrieve the contents of user spaces, and change and retrieve information about user spaces.

For additional information, see Using User Space APIs.

The user space APIs are:

- Change User Space (QUSCHGUS) changes the contents of a user space.
- Change User Space Attributes (QUSCUSAT) changes the attributes of a user space object.
- Create User Space (QUSCRTUS) creates a user space.
- Delete User Space (QUSDLTUS) deletes user spaces created with the QUSCRTUS API.
- Retrieve Pointer to User Space (QUSPTRUS) retrieves a pointer to the beginning of a user space for a high-level language (HLL) that supports pointers. HLLs that support pointers can use this pointer to manipulate the contents of a user space directly.
- Retrieve User Space (QUSRTVUS) retrieves the contents of a user space. It does not retrieve descriptive information about the user space, such as its size.
- Retrieve User Space Attributes (QUSRUSAT) retrieves information about creation attributes and current operational statistics of the user space, such as its size.

Using User Space APIs

User spaces are objects that consist of a collection of bytes used for storing user-defined information. They are permanent objects that are located in either the system domain or the user domain. They have an object type of *USRSPC and a maximum size of 16MB. You can save and restore user spaces to other systems. If, however, the user spaces contain pointers, you cannot restore the pointers even if you want to restore them to the same system.

You can use the user space APIs to:

- Create user spaces to be used by list APIs to generate lists of data.
- Store pointers.
- Store large amounts of data. You can create a user space as large as 16 megabytes. You cannot create a data area larger than 2000 bytes.
- Save information in user space objects, and save and restore the object with the information in it using CL commands.
- Pass data from job to job or from system to system.

Note: If the allow user domain (QALWUSRDMN) system value contains only the QTEMP library, you can only use the user space through APIs unless the user space is in QTEMP. You cannot use the Retrieve Pointer to User Space API.

Top | Object APIs | APIs by category

Change User Space (QUSCHGUS) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Starting position	Input	Binary(4)
3	Length of data	Input	Binary(4)
4	Input data	Input	Char(*)
5	Force changes to auxiliary storage	Input	Char(1)

Optional Parameter:

6	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Change User Space (QUSCHGUS) API changes the contents of the user space (*USRSPC) object by moving a specified amount of data to the object. This API allows you to change the contents of a user space if you are using either:

- A language that does not support pointers
- System-domain user spaces

Note: To determine the starting position for the QUSCHGUS API, you must add 1 to the offset value. In contrast to the OS/400 list APIs, which use an offset value based on 0 for the starting position, the QUSCHGUS API uses a value based on 1. For the QUSCHGUS API, the first character in the user space is at position 1.

Authorities and Locks

Library Authority

*EXECUTE

User Space Authority

*CHANGE

User Space Lock

*EXCLRD

Required Parameter Group

Qualified user space name

INPUT; CHAR(20)

The first 10 characters contain the user space name, and the second 10 characters contain the name of the library where the user space is located. The special values supported for the library name are *LIBL and *CURLIB.

Starting position

INPUT; BINARY(4)

The first byte of the user space that is to be changed. It must have a value greater than 0.

Length of data

INPUT; BINARY(4)

The length of the new data in the input data parameter. The length must be greater than 0.

Input data

INPUT; CHAR(*)

The new data to be placed into the user space. The field must be at least as long as the length of data parameter.

Force changes to auxiliary storage

INPUT; CHAR(1)

The method of forcing changes made to the user space to auxiliary storage.

The valid values are as follows:

0 Does not force changes. Normal system management writes the changes to auxiliary storage.

1 Forces changes asynchronously. This interrupts the normal system management and ensures that the user space is written to auxiliary storage.

2 Forces changes synchronously. This interrupts the normal system management and ensures that the user space is written immediately to auxiliary storage.

Optional Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C0F E	Value &1 for starting position parameter is not valid.
CPF3C04 E	User space &1 not changed.
CPD3C0F D	Value &1 for starting position parameter is not valid.
CPD3C12 D	Length of data is not valid.
CPD3C13 D	Value &1 for force option is not valid.
CPD3C14 D	Starting position &1 and length &2 cause space overflow.
CPD3C15 D	New value &1 is shorter than the length specified.
CPD3C17 D	Error occurred with input data parameter.
CPF3C12 E	Length of data is not valid.
CPF3C13 E	Value &1 for force option is not valid.
CPF3C14 E	Starting position &1 and length &2 cause space overflow.
CPF3C15 E	New value &1 is shorter than the length specified.
CPF3C17 E	Error occurred with input data parameter.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
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.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V1R3

Top | Object API categories | API by category

Change User Space Attributes (QUSCUSAT) API

Required Parameter Group:

1	Returned library name	Output	Char(10)
2	Qualified user space name	Input	Char(20)
3	Attributes to change	Input	Char(*)
4	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Change User Space Attributes (QUSCUSAT) API changes the attributes of a user space object. This API can be used to:

- Extend or truncate a user space
- Mark or unmark the user space as automatically extendible by the system
- Change the initial value to which future extensions of the user space will be set
- Change the number of pages transferred between main storage and auxiliary storage

Authorities and Locks

Library Authority

*EXECUTE

User Space Authority

*CHANGE, *OBJMGT

User Space Lock

*EXCL

Required Parameter Group

Returned library name

OUTPUT; CHAR(10)

The name of the library that contains the changed user space object. If the space attributes are successfully changed, the name of the library in which the user space was found is returned.

Qualified user space name

INPUT; CHAR(20)

The first 10 characters contain the user space name, and the second 10 characters contain the name of the library where the user space is located. The special values supported for the library name are *LIBL and *CURLIB.

Attributes to change

INPUT; CHAR(*)

The attributes of the user space object that you want to change.

The information must be in the following format:

Number of variable length records	BINARY(4) The total number of all of the variable length records.
Variable length records	The attributes of the user space to change and the data used for the change. For the specific format of the variable length record, refer to Format for Variable Length Records.

Error code

I/O; CHAR(*)

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

Format for Variable Length Records

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

If you specify a length of data that is longer than the key field's defined data length, the data will be truncated at the right. No error message will be returned.

If you specify a length of data that is shorter than the key field's defined data length, an error message will be returned.

You may specify a key more than once. If duplicate keys are specified, the last specified value for that key is used.

Each variable length record must be 4-byte aligned. If not, unpredictable results may occur.

Field Descriptions

Data. The value to which a specific user space attribute is to be changed. All values are validity checked.

Key. The user space attribute to be changed. Only specific attributes can be changed. Refer to Keys for more information.

Length of data. The length of the new user space attribute value. The length of data field is used to get addressability to the next attribute record.

Keys

The following table lists the keys that can be used in the attribute record.

Key	Type	Attribute
1	BINARY(4)	Space size
2	CHAR(1)	Initial value
3	CHAR(1)	Automatic extendibility
4	BINARY(4)	Transfer size request

Field Descriptions

Automatic extendibility. Whether or not the user space is automatically extended by the system when the end of the space is encountered.

0 The user space is not automatically extendible.

1 The user space is automatically extendible.

Initial value. The initial value to which future extensions of the user space will be set. You will achieve the best performance if you set this byte to hexadecimal zeros (X'00').

Space size. The size in bytes of the user space object. If this value is smaller than the current size of the space, the user space is truncated. If it is larger, the space is extended.

Transfer size request. The number of pages to be transferred between main storage and auxiliary storage. This is only a request, as the machine may use a value of its choice in some circumstances. Allowable values range between 0 and 32 pages. A value of 0 is an indication that the machine should use the default transfer size for the user space. A larger transfer size may allow for better performance of applications processing the user space.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C4B E	Value not valid for field &1.
CPF3C4C E	Value not valid for field &1.
CPF3C4D E	Length &1 for key &2 not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
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.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Top | Object API categories | API by category

Create User Space (QUSCRTUS) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Extended attribute	Input	Char(10)
3	Initial size	Input	Binary(4)
4	Initial value	Input	Char(1)
5	Public authority	Input	Char(10)
6	Text description	Input	Char(50)

Optional Parameter Group 1:

7	Replace	Input	Char(10)
8	Error code	I/O	Char(*)

Optional Parameter Group 2:

9	Domain	Input	Char(10)

Optional Parameter Group 3:

10	Transfer size request	Input	Binary(4)
11	Optimum space alignment	Input	Char(1)

Default Public Authority: *USE

Threadsafe: Yes

The Create User Space (QUSCRTUS) API creates a user space in either the user domain or the system domain. A system-domain user space cannot be saved to a release prior to Version 2 Release 3 Modification 0. A user-domain user space can be directly manipulated with machine interface (MI) instructions or can be accessed using system APIs. On systems with a QSECURITY system value of 40 or greater, applications can only access system-domain user spaces using APIs. The user space objects you create are larger than or equal to the size specified. They have a fixed length and can be extended or truncated using the Change User Space Attributes (QUSCUSAT) API of the Modify Space (MODS) MI instruction (for user-domain user spaces). (The MODS instruction will not work on system-domain user spaces if the security level of the system is 40 or greater.)

Note: For performance reasons, the *USRSPC object is created before checking to see if it exists in the library specified for the qualified user space name. If you have an application using this API repeatedly, even if you are using *NO for the replace parameter, permanent system addresses will be used. Change User Space Attributes (QUSCUSAT) API or the Modify Space (MODS) MI instruction (for user-domain user spaces). (The MODS instruction will not work on system-domain user spaces if the security level of the system is 40 or greater.)

Authorities and Locks

User Space Authority

*OBJMGT, *OBJEXIST, and *READ. These authorities are required only if the replace parameter is used and if there is an existing user space to replace.

User Space Library Authority

*READ and *ADD.

User Space Lock

*EXCL. This applies to both the user space being created and an existing user space being replaced.

Required Parameter Group

Qualified user space name

INPUT; CHAR(20)

The first 10 characters contain the user space name, and the second 10 characters contain the name of the library where the user space is located. The only special value supported for the library name is *CURLIB.

User spaces created in the QTEMP and QRPLOBJ libraries are not forced to permanent storage; they are deleted when those libraries are cleared at sign-off and system IPL, respectively.

Extended attribute

INPUT; CHAR(10)

The extended attribute of the user space. For example, an object type of *FILE has an extended attribute of PF (physical file), LF (logical file), DSPF (display file), SAVF (save file), and so on.

The extended attribute must be a valid *NAME. You can enter this parameter in uppercase, lowercase, or mixed case. The API converts it to uppercase.

Initial size

INPUT; BINARY(4)

The initial size of the user space being created. This value must be from 1 byte to 16, 776, 704 bytes.

Initial value

INPUT; CHAR(1)

The initial value of all bytes in the user space. You will achieve the best performance if you set this byte to X'00'.

Public authority

INPUT; CHAR(10)

The authority you give users who do not have specific private or group authority to the user space. Once the user space has been created, its public authority stays the same when it is moved to another library or restored from backup media.

If the replace parameter is used and a user space exists to be replaced, this parameter is ignored. All authorities are transferred from the replaced user space to the new one.

The valid values for this parameter are:

**ALL*	The user can perform all authorized operations on the object.
Authorization list name	The user space is secured by the specified authorization list, and its public authority is set to *AUTL. The specified authorization list must exist on the system when this API is called. If it does not exist, the create process fails, and an error is returned to the application.
**CHANGE*	The user can read the object description and has read, add, update, and delete authority to the object.
**EXCLUDE*	The user cannot access the object in any way.
**LIBCRTAUT*	The public authority for the user space is taken from the CRTAUT value for the target library when the object is created. If the CRTAUT value for the library changes later, that change does not affect user spaces already created. If the CRTAUT value contains an authorization list name and that authorization list secures an object, do not delete the list. If you do, the next time you call this API with the *LIBCRTAUT parameter, it will fail.
**USE*	The user can read the object and its description but cannot change them.

Text description

INPUT; CHAR(50)

This text briefly describes the user space.

Optional Parameter Group 1

Replace

INPUT; CHAR(10)

Whether you want to replace an existing user space.

Valid values for this parameter are:

**NO* Do not replace an existing user space of the same name and library. *NO is the default value.

**YES* Replace an existing user space of the same name and library.

If the user space already exists, it is replaced by a new user space of the same name and library, and is subject to the same authorities. The user space being replaced is destroyed if both:

- The allow user domain (QALWUSRDMN) system value is not set to *ALL or does not contain the library QRPLOBJ.
- The user space you are replacing is in the user domain.

If the user space is in the system domain, it is moved to QRPLOBJ. If QALWUSRDMN is set to *ALL or if it contains QRPLOBJ, the replaced user space is moved to QRPLOBJ, which is cleared at system IPL. For details about authorities, ownership, and renaming, see the discussion of the REPLACE parameter in the Control Language (CL) information in the iSeries Information Center.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Optional Parameter Group 2

Domain

INPUT; CHAR(10)

The domain into which the user space is created. If this parameter is not specified, the value of *DEFAULT is assumed by the API.

Valid values for this parameter are:

*DEFAULT	Allows the system to decide into which domain the object should be created.
*SYSTEM	Creates the user space object into the system domain. The API can always create a user space into the system domain regardless of the security level in effect. However, you must use APIs to access system-domain user spaces if you are running at security level 40 or greater.
*USER	Attempts to create the user space object into the user domain. This is not always possible. If the library you are creating the user space into does not appear in the QALWUSRDMN system value, the API cannot create the user space into the user domain. An error will be returned.

The API uses the following values to determine into which domain to create the user space. The destination library is the library you specified in the qualified user space name parameter. The optional domain parameter is the information specified in the domain parameter.

QALWUSRDMN System Value	Destination Library	Optional Domain Parameter	Domain of Created Object
*ALL	Any	*DEFAULT	User domain
*ALL	Any	*SYSTEM	System domain
*ALL	Any	*USER	User domain
QTEMP	QTEMP	*DEFAULT	User domain
QTEMP	QTEMP	*SYSTEM	System domain
QTEMP	QTEMP	*USER	User domain

Does not contain library name	Library name	*DEFAULT	System domain
Does not contain library name	Library name	*SYSTEM	System domain
Does not contain library name	Library name	*USER	None; error is returned
Note: The QALWUSRDMN system value lists the libraries into which user domain objects can be created. The libraries can be the special value *ALL or a list of one or more library names.			

You must use APIs to access data or information in system-domain user spaces on systems with a QSECURITY level of 40 or greater. You cannot use MI instructions to directly access system-domain user objects.

The Retrieve Object Description (QUSROBJD) or List Objects (QUSLOBJ) API can be used to determine into which domain the user-space object was created.

Optional Parameter Group 3

Transfer size request

INPUT; BINARY(4)

The number of pages to be transferred between main storage and auxiliary storage This is only a request, as the machine may use a value of its choice in some circumstances. Allowable values range between 0 and 32 pages. A value of 0 is an indication that the machine should use the default transfer size for the user space. If this parameter is not specified, the default is 0. A larger transfer size may allow for better performance of applications processing the user space.

Optimum space alignment

INPUT; CHAR(1)

Allows the machine to choose optimum alignment for the user space. Optimum alignment may allow for better performance of applications that manipulate the user space.

Allowable values are:

0 Do not choose optimum space alignment. 0 is the default value.

1 Choose optimum space alignment.

Note: If not using the optimum space alignment, the user space has a maximum size of 16MB minus 512 bytes (16,776,704 bytes). If optimum alignment is specified, the maximum size of the user space is 16MB minus one disk page (current page size is 4096 bytes, giving a maximum space size of 16,773,120 bytes).

Error Messages

Message ID	Error Message Text
CPF2143 E	Cannot allocate object &1 in &2 type *&3.
CPF2144 E	Not authorized to &1 in &2 type *&3.
CPF2283 E	Authorization list &1 does not exist.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C01 E	User space &2 in library &1 not created.
CPD3C01 D	Object name &1 is not valid.
CPD3C03 D	Extended attribute &1 is not valid.
CPD3C04 D	Value &1 for size parameter is not valid.
CPD3C05 D	Value &1 for authority parameter is not valid.
CPF3C2B E	Extended attribute &1 is not valid.
CPF3C2C E	Value &1 for size parameter is not valid.
CPF3C2D E	Value &1 for authority parameter is not valid.
CPF3C29 E	Object name &1 is not valid.
CPF3C34 E	Value &1 for replace option is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C45 E	Value &1 not valid for domain parameter.
CPF3C49 E	Request for user domain object cannot be granted.
CPF3C90 E	Literal value cannot be changed.
CPF3C91 E	Value &1 not valid for transfer size request parameter.
CPF3C92	Value &1 not valid for optimum space alignment parameter.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9838 E	User profile storage limit exceeded.
CPF9870 E	Object &2 type *&5 already exists in library &3.

CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V1R3

Top | Object API categories | API by category

Delete User Space (QUSDLTUS) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Delete User Space (QUSDLTUS) API deletes user spaces created with the Create User Space (QUSCRTUS) API. The QUSDLTUS API performs the same function as the Delete User Space (DLTUSRSPC) command.

Authorities and Locks

Library Authority

*EXECUTE

User Space Authority

*OBJEXIST

User Space Lock

*EXCL

Required Parameter Group

Qualified user space name

INPUT; CHAR(20)

The name of the user space and the name of the library in which it resides. The first 10 characters contain the user space name, and the second 10 characters contain the library name.

The user space name can be either a specific name or a generic name, a string of one or more characters followed by an asterisk (*). If you specify a generic name, QUSDLTUS deletes all user spaces that have names beginning with the string for which the user has authority.

You can use these special values for the library name:

**ALL*	All libraries
**ALLUSR*	All user-defined libraries, plus libraries containing user data and having names starting with Q. For information on the libraries included, see *ALLUSR.
**CURLIB*	The job's current library

**LIBL* The library list

**USRLIBL* The user portion of the job's library 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
CPF2105 E	Object &1 in &2 type *&3 not found.
CPF2110 E	Library &1 not found.
CPF2113 E	Cannot allocate library &1.
CPF2114 E	Cannot allocate object &1 in &2 type *&3.
CPF2117 E	&4 objects type *&3 deleted. &5 objects not deleted.
CPF2125 E	No objects deleted.
CPF2176 E	Library &1 damaged.
CPF2182 E	Not authorized to library &1.
CPF2189 E	Not authorized to object &1 in &2 type *&3.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R1

Retrieve Pointer to User Space (QUSPTRUS) API

Required Parameter Group:			
1	Qualified user space name	Input	Char(20)
2	Return pointer	Output	PTR(SPP)
Optional Parameter:			
3	Error code	I/O	Char(*)
Default Public Authority: *USE			
Threadsafe: Yes			

The Retrieve Pointer to User Space (QUSPTRUS) API retrieves a pointer to the contents of a user-domain user space. The data in that user space then can be directly manipulated by high-level language programs that support pointers, such as C or COBOL. The QUSPTRUS API will not return a pointer to a system-domain user space; you must use system APIs to access system-domain user spaces. If you attempt to retrieve the pointer to a system-domain user space, an error will be returned.

The QUSPTRUS API even returns a pointer to an object that is subject to an exclusive (*EXCL) lock. If you create application programs using HLLs that can directly update user spaces using pointers (instead of using the Change User Space (QUSCHGUS) API), you should use your own synchronization data methods. You can use one of the following methods to avoid updates at the same time to the same location within a user space:

- CMPSW MI instruction
- CMPSWP MI instruction
- LOCK MI instruction
- LOCKSL MI instruction
- Allocate Object (ALCOBJ) command

Use of the QUSPTRUS API does not update the object usage information (such as last changed date, last date used, and so on). You should use the Change User Space or the Retrieve User Space API to update the object usage information if needed.

Examples of the API are in API examples.

Authorities and Locks

Library Authority

*EXECUTE

User Space Authority

*USE

Required Parameter Group

Qualified user space name

INPUT; CHAR(20)

The first 10 characters contain the user space name, and the second 10 characters contain the name of the library where the user space is located. The special values supported for the library name are *LIBL and *CURLIB.

Return pointer

OUTPUT; PTR(SPP)

The variable containing the pointer to the user space after the QUSPTRUS API has completed running. This parameter must be on a 16-byte boundary alignment.

Optional Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C05 E	One or more errors found while trying to retrieve a pointer.
CPF3C18 E	Pointer parameter is not on a 16-byte boundary.
CPD3C18 D	Pointer parameter is not on a 16-byte boundary.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C48 E	Operation not valid on system domain object.
CPF3C90 E	Literal value cannot be changed.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
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.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Top | Object API categories | API by category

Retrieve User Space (QUSRTVUS) API

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Starting position	Input	Binary(4)
3	Length of data	Input	Binary(4)
4	Receiver variable	Output	Char(*)

Optional Parameter Group:

5	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve User Space (QUSRTVUS) API allows you to retrieve the contents of a user space. The QUSRTVUS API does not retrieve descriptive information about the user space object, such as its size. To retrieve information about the attributes of a user space, refer to the Retrieve User Space Attributes (QUSRUSAT) API.

If you are repeatedly accessing the contents of a user space and are using an HLL that supports pointers, see the Retrieve Pointer to User Space (QUSPTRUS) API; this API provides a pointer to the user space for improved performance. When you have obtained a pointer, you use pointer arithmetic to access the contents of a user space.

Note: To determine the starting position for the QUSRTVUS API, you must add 1 to the offset value. In contrast to the OS/400 list APIs, which use an offset value based on 0 for the starting position, the QUSRTVUS API uses a value based on 1. For the QUSRTVUS API, the first character in the user space is at position 1.

Authorities and Locks

Library Authority

*EXECUTE

User Space Authority

*USE

User Space Lock

*SHRNUP

Required Parameter Group

Qualified user space name

INPUT; CHAR(20)

The first 10 characters contain the user space name, and the second 10 characters contain the name of the library where the user space is located. The special values supported for the library name are *LIBL and *CURLIB.

Starting position

INPUT; BINARY(4)

The first byte of the user space to be retrieved. A value of 1 will identify the first character in the user space.

Length of data

INPUT; BINARY(4)

The length of the data to retrieve. This length must not be larger than the size of the variable that is to receive the data. It must also be greater than 0.

Receiver variable

OUTPUT; CHAR(*)

The variable that will receive the contents of the user space being retrieved.

Optional Parameter

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C0F E	Value &1 for starting position parameter is not valid.
CPF3C06 E	Information not retrieved from user space &1.
CPD3C0F D	Value &1 for starting position parameter is not valid.
CPD3C12 D	Length of data is not valid.
CPD3C14 D	Starting position &1 and length &2 cause space overflow.

CPD3C16 D	Receiver area too small for length of data &1 specified.
CPD3C20 D	Error occurred with receiver variable specified.
CPF3C12 E	Length of data is not valid.
CPF3C14 E	Starting position &1 and length &2 cause space overflow.
CPF3C16 E	Receiver area too small for length of data &1 specified.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
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.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V1R3

Top | Object API categories | API by category

Retrieve User Space Attributes (QUSRUSAT) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Qualified user space name	Input	Char(20)
5	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve User Space Attributes (QUSRUSAT) API retrieves information about the current attributes and the current operational statistics of the user space.

You can also retrieve information about user space attributes by using one of the following:

- The Materialize Space (MATS) machine interface (MI) instruction
- The Retrieve Object Description (QUSROBJD) API described on page
- The Retrieve Object Description (RTVOBJD) command

Authorities and Locks

User Space Library Authority

*EXECUTE

User Space Authority

*USE

User Space Lock

*SHRNUP

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable that is to receive the information requested. You can specify the size of this 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 may not be predictable. The minimum length is 8 bytes.

Format name

INPUT; CHAR(8)

The format of the space information to be returned.

The format names supported are:

SPCA0100 Basic information

Refer to SPCA0100 Format for details on the format.

Qualified user space name

INPUT; CHAR(20)

The user space for which you want to retrieve information, and the library in which it is located. The first 10 characters contain the user space name, and the second 10 characters contain the library name.

You can use these special values for the library name:

**CURLIB* The job's current library

**LIBL* The library list

Error code

I/O; CHAR(*)

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

SPCA0100 Format

The following information about a user space is returned for the SPCA0100 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Space size
12	C	CHAR(1)	Automatic extendibility
13	D	CHAR(1)	Initial value
14	E	CHAR(10)	User space library name

Field Descriptions

Automatic extendibility. Whether or not the space is extended automatically by the system when the end of the space is encountered.

0 Space is not automatically extendible

1 Space is automatically extendible

Bytes available. The length of all data available to return. All available data is returned if enough space is provided.

Bytes returned. The length of the data actually returned.

Initial value. The initial value to which future extensions of the user space will be set.

Space size. The size of the user space object in bytes.

User space library name. The library in which the user space is located. This is helpful when *LIBL or *CURLIB is specified as the library name in the qualified user space name parameter.

Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
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.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.

CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Top | Object API categories | API by category

Object-related APIs

You can use object-related APIs to obtain information about OS/400 objects.

The QUSLOBJ and QUSROBJD APIs return much of the same information. The APIs differ, however, in several respects:

- The APIs group the returned information differently among their output formats. For example, the OBJL0300 format of the QUSLOBJ API does not contain exactly the same data as the OBJD0300 format of the QUSROBJD API.
- The APIs use different data formats for some specific items, such as dates and times.
- The APIs differ in efficiency, depending on your application. In most cases, the QUSROBJD API is faster at retrieving information about a single object. The QUSLOBJ API is faster at retrieving information about several objects.

The object-related APIs are:

- Change Library List (QLICHGLL) changes the current library, the two product libraries, and the user part of the job's library list.
- Change Object Description (QLICOBJD) changes object information for a specific object.
- Convert Type (QLICVTTP) converts an object type to and from hexadecimal format.
- List Objects (QUSLOBJ) generates a list of object names and descriptive information based on the specified parameters.
- Materialize Context (QusMaterializeContext) returns either the type and subtype of the object or system pointers for all or for a selected set of objects that are contained by the context.
- Move Folder to ASP (QHSMMOVF) moves a root folder and its contents from its existing auxiliary storage pool (ASP) to the specified target ASP through a save and restore process.
- Move Library to ASP (QHSMMOVL) moves a library and its contents from its existing auxiliary storage pool (ASP) to the specified target ASP through a save and restore process.
- Open List of Objects (QGYOLOBJ) generates a list of object names and descriptive information based on specified selection parameters.
- Rename Object (QLIRNMO) renames an existing object to a new object name or new library name or both and optionally replaces the object.
- Retrieve Library Description (QLIRLIBD) retrieves attributes for a specific library.
- Retrieve Object Description (QUSROBJD) retrieves object information for a specific object.

Change Library List (QLICHGLL) API

Required Parameter Group:

1	Current library name	Input	Char(11)
2	First product library name	Input	Char(11)
3	Second product library name	Input	Char(11)
4	User library list names	Input	Array(*) of Char(11)
5	Number of user library names	Input	Binary(4)
6	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Change Library List (QLICHGLL) API changes the current library, the two product libraries, and the user part of the current thread's library list.

When the initial thread's library list is changed, each library added to the list may be locked with a shared-read lock. The value of the QLIBLCKLVL system value determines whether libraries in the library list are locked. If a library is being removed from the initial thread's library list, and it was locked when it was added to the library list, the shared-read lock is released. When a secondary thread's library list is changed, libraries added to the library list are not locked.

Authority to the library is checked whenever a library name is specified on one of the required parameters.

You may want to save the current thread's library list before changing it. The QWCRTVCA API can be used to retrieve the current thread's library list. The Retrieve Job Information (QUSRJOBI) API can be used to retrieve the initial thread's library list, when the current thread is not the initial thread. You can then change the library list back to its original value by using the QLICHGLL API with the libraries saved from the QUSRJOBI or QWCRTVCA API.

Authorities and Locks

Library Authority

*USE

Library Lock

*SHRRD (initial thread only)

The QLIBLCKLVL system value determines whether libraries in the library list are locked.

Required Parameter Group

Current library name

INPUT;CHAR(11)

The library that replaces the current library in the library list. (The data is left-justified with a blank at the end.)

The current library can be, but does not have to be, a duplicate of any library in the library list. QTEMP cannot be specified for the library name.

The following special values can be used:

**CRTDFT* No library should be in the current library entry in the library list. If objects are created into the current library, then library QGPL is used as the current default library.

**SAME* The current library in the library list is not changed.

First product library name

INPUT;CHAR(11)

The library that replaces the first product library entry in the library list. (The data is left-justified with a blank at the end.)

A product library may be a duplicate of the current library or of a library in the user part of the library list. QTEMP cannot be specified for the library name.

The following special values can be used:

**NONE* The first product library entry in the library list is removed.

**SAME* The first product library entry in the library list is not changed.

Second product library name

INPUT;CHAR(11)

The library that replaces the second product library entry in the library list. (The data is left-justified with a blank at the end.) QTEMP cannot be specified for the library name.

The following special values can be used:

**NONE* The second product library entry in the library list is removed.

**SAMET* he second product library entry in the library list is not changed.

User library list names

INPUT; ARRAY(*) of CHAR(11)

The libraries that replace the libraries in the user part of the library list. Specify the names of the libraries in the order in which they are to be searched. (The data is left-justified with a blank at the end.)

The same library name cannot be specified more than once. A library cannot exist in the system part and the user part of the library list.

Number of user library names

INPUT;BINARY(4)

The total number of library names that will be changed in the user part of the library list.

This must be a value from -1 through 250. There is a limit of 250 libraries in the user part of the library list. When either of the following values is specified, the user library list names parameter is ignored:

-1 The user part of the library list remains the same.

0 All libraries in the user part of the library list are removed.

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
CPF2106 E	Library list not changed.
CPF2110 E	Library &1 not found.
CPF2113 E	Cannot allocate library &1.
CPF2133 E	First product library on library list destroyed.
CPF2134 E	Second product library on library list destroyed.
CPF2137 E	Current library on library list destroyed.
CPF2176 E	Library &1 damaged.
CPF2182 E	Not authorized to library &1.
CPF2184 E	Library list not replaced.
CPF219A E	Library QTEMP cannot be specified.
CPF219F E	Number of libraries for library list not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.

CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R1

Top | Object API categories | API by category

Change Object Description (QLICOBJD) API

Required Parameter Group:

1	Returned library name	Output	Char(10)
2	Object and library name	Input	Char(20)
3	Object type	Input	Char(10)
4	Changed object information	Input	Char(*)
5	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Change Object Description (QLICOBJD) API lets you change object information for a specific object.

Before the object can be changed with the API, the allow change by program field is checked. If the API cannot be used to change the object, message CPF219B is issued.

When an object has been successfully updated by the API, the date and time the object was changed and the changed by program field are updated.

Authorities and Locks

Library Authority

*EXECUTE

*Non-*FILE Object Authority*

*OBJMGT

**FILE Object Authority*

*OBJOPR and *OBJMGT

Object Lock

*EXCLRD

Required Parameter Group

Returned library name

OUTPUT; CHAR(10)

The name of the library that contains the changed object. If *CURLIB, *LIBL, or a name is specified for the library name in the object and library name parameter, the value returned is the name of the library where the object was found. If an error occurred while the API was accessing the object, blanks are returned.

Object and library name

INPUT; CHAR(20)

The object for which you want to change information and the library in which it is located. The first 10 characters contain the object name, and the second 10 characters contain the library name. You can use these special values for the library name:

**CURLIB*	The job's current library
**LIBL*	The job's library list

Object type

INPUT; CHAR(10)

The type of object for which you want to change the information. You can only specify specific external object types. An asterisk (*) must precede the object type. For a complete list of the available object types, see the Control Language (CL) information.

Changed object information

INPUT; CHAR(*)

The information for the object that you want to change. The information must be in the following format:

Number of variable length records	BINARY(4) Total number of all of the variable length records. If the value of this field is less than 0, an error message is returned.
Variable length records	The fields of the object's description to change and the data used for the change. For the specific format of the variable length record, see Format for Variable Length Record.

Error code

I/O; CHAR(*)

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

Format for Variable Length Record

The following table defines the format for the variable length records.

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

If the length of the data is longer than the key field's data length, the data will be truncated at the right. No message will be issued.

If the length of the data is smaller than the key field's data length, the data will be padded with blanks at the right. No message will be issued.

It is not an error to specify a key more than once. If duplicate keys are specified, the last specified value for that key is used.

Each variable length record must be 4-byte aligned. If not, unpredictable results may occur.

Field Descriptions

Data. The data used to change a specific field of the object description. Validity checking is done on the values specified for the following keys to verify that they are 0 or 1.

- Allow change by program
- Days used count
- Last used date
- Changed date and time stamp

The data specified for other keys is not validity checked.

Key. Identifies a field of the object's description to change. Only specific fields of the object's description can be changed. See Keys for the list of valid keys.

Length of data. The length of the data used to change a specific field of the object's description. If the value of this field is 0 or negative, an error message is returned.

Keys

The following table lists the valid keys for the key field area of the variable length record.

Key	Type	Field
1	CHAR(30)	Source file
2	CHAR(13)	Source file last changed date and time
3	CHAR(13)	Compiler
4	CHAR(8)	Object control level
5	CHAR(13)	Licensed program
6	CHAR(7)	Program temporary fix (PTF)
7	CHAR(6)	Authorized program analysis report (APAR)
8	CHAR(1)	Allow change by program
9	CHAR(10)	User-defined attribute
10	CHAR(50)	Text
11	CHAR(1)	Days used count
12	CHAR(4)	Product option load ID
13	CHAR(4)	Product option ID

14	CHAR(4)	Component ID
15	CHAR(1)	Last used date
16	CHAR(1)	Changed date and time stamp
17	CHAR(10)	Member's days used count

Field Descriptions

Allow change by program. Whether to prevent users from changing an object's description with this API. It must have a value of 0 or 1.

0 Changes are not allowed with the API. Once this field has been changed to 0, the API cannot be used to make any further changes to the object's description.

1 Changes are allowed with the API. The API can be used to change the object's description.

Authorized program analysis report (APAR). The authorized program analysis report identification that caused this object to be patched. IBM APARs have an uppercase alphabetic character followed by 5 decimal numbers. If you want to conform with the system, this format should be followed.

Changed date and time stamp. The date and time the object was last changed. This is useful if you only want to update an object's change date/time stamp and do not want to update any other key fields.

0 The changed date and time stamp is not updated.

1 The changed date and time stamp is updated to the current system date and time.

This key cannot be specified with any other key. If it is, message CPF21A6 is issued.

Note: For database files, the last change date/time is updated for all members in the file and the file itself.

Compiler. The name, version level, release level, and modification level of the compiler. Objects created with IBM products will have the licensed program name of the compiler in the compiler name field and a version field in the VxRxMy format where x must be 0-9 and y must be 0-9 or A-Z. If you want to conform with the system, this format should be followed.

Compiler name CHAR(7)

Version CHAR(6)

Component ID. The product administrator owns this field. It can be used to track information about objects, such as object size, at a lower level than the product option ID.

Days used count. This field is used to:

- Reset the number of days an object has been used on the system.
- Update the date the days used count was last reset to 0.

It must have a value of 0 or 1.

0 Neither the days used count nor the reset date on the Display Object Description panel is updated.

1 The days used count is set to 0. The reset date on the Display Object Description panel is updated to the current system date.

Key fields 11 and 15 cannot both be specified. These changes are incompatible because key 11 will change the days used count to 0 and key 15 will increase the days used count. If both keys 11 and 15 are specified, error message CPF21A1 is issued.

This field cannot be specified for *DOC object types or when object usage information is not updated for this object type. Object usage information is not updated for all object types. For more details on usage information, see the CL Programming book. Use the **Qp0lSetAttr()** API to update this field for *DOC object types. If this field is not allowed for an object type, error message CPF2131 is issued.

Note: For database files, the days used count reset date and the days used count are updated for all members in the file.

Last used date. This field is used to:

- Update the last used date to the current system date.
- Increase the days used count.

It must have a value of 0 or 1.

0 The last used date field is not updated. The field is not increased.

1 The last used date field is updated to the current system date. If this is the first use of the object today (since midnight), the days used count field is increased.

This field cannot be specified for *DOC object types or when object usage information is not updated for this object type. Object usage information is not updated for all object types. For more details on usage information, see the CL Programming book. Use the **Qp0lSetAttr()** API to update this field for *DOC object types. If this field is not allowed for an object type, error message CPF2131 is issued.

Note: For database files, the last used date and number of days used count are updated for all members in the file.

The last used date field cannot be changed for a database file that does not have any members. Error message CPF21A2 will be issued.

Licensed program. The name, version level, release level, and modification level of the licensed program. Objects that are a part of an IBM licensed program have a valid licensed program name (7 characters containing 0-9 and uppercase A-Z). The version field is in the VxRxMy format where x must be 0-9 and y must be 0-9 or A-Z. If you want to conform with the system, this format should be followed.

Licensed program name	CHAR(7)
Version	CHAR(6)

Member's days used count. This field is used to:

- Reset the number of days a database file member has been used on the system.
- Update the member's date that the days used count was last reset to 0.

This field must be a valid 10-character file-member name and must be padded with blank characters. No special values are allowed.

It must be a valid 10-character file-member name and must be padded with blank characters. No special values are allowed.

Key fields 11 and 17 cannot both be specified. These changes are incompatible because key 11 will change the days used count for all members in a file and key 17 will change the days used count for a single member. If both keys 11 and 17 are specified, error message CPF21A1 is issued.

Key fields 15 and 17 cannot both be specified. These changes are incompatible because key 15 will increase the days used count and key 17 will change the days used count to 0 for a member. If both keys 15 and 17 are specified, error message CPF21A1 is issued.

This field can be specified only for an object type of *FILE. If the object is not a *FILE object, error message CPF2131 is issued.

Note: This field updates the days used count for a single member only. To reset the days used count for all members in a database file, specify key field 11.

Object control level. The object control level for the object. IBM programs will have an 8-character decimal value.

Product option ID. Identifies part of a licensed program (product). Products can have multiple options. Objects that are a part of an IBM licensed program must be 0000 (*BASE) through 0099. If you want to conform with the system, this format should be followed.

Product option load ID. The language identifier associated with the object. Objects that are a part of an IBM licensed program must have one of the allowed languages in the 29xx format. If you want to conform with the system, this format should be followed.

Program temporary fix. The program temporary fix (PTF) that resulted in the creation of the object. For IBM objects the first 2 characters are a prefix ID, and the remaining 5 characters are the program change ID (decimal). The field is blank if the object was not changed because of a PTF. If you want to conform with the system, this format should be followed.

Source file. The source file used to create the object, the library name in which it is located, and the name of the source file member.

Source file name	CHAR(10)
Library name	CHAR(10)
Member name	CHAR(10)

Objects created with IBM products have valid object names for the qualified source file name. If you want to conform with the system, this format should be followed.

Source file last changed date and time. The date and time the member in the source file was last updated. Objects created with IBM products will be in the CYYMMDDHHMMSS format:

C	Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
YY	Year
MM	Month
DD	Day
HH	Hour

MM Minute

SS Second

If you want to conform with the system, this format should be followed.

Text. The user-defined text that briefly describes the object and its function.

User-defined attribute. An attribute you define. This should not be confused with the extended attribute of the object. The extended attribute is set by the system when an object is created.

Error Messages

Message ID	Error Message Text
CPF21A1 E	Key &1 not allowed with key &2.
CPF21A2 E	Last used date for &1 in &2 type *FILE cannot be changed.
CPF21A6 E	Cannot specify field &1 with any other fields.
CPF2131 E	Key &1 not allowed with object type *&2.
CPF2150 E	Object information function failed.
CPF2151 E	Operation failed for &2 in &1 type *&3.
CPF219B E	Cannot change &1 in &2 type *&3.
CPF219E E	Object type *&1 not valid external object type.
CPF2199 E	&2 not valid for field &1.
CPF24B4 E	Severe error while addressing parameter list.
CPF2451 E	Message queue &1 is allocated to another job.
CPF3CF1 E	Error code parameter not valid.
CPF3C4D E	Length &1 for key &2 not valid.
CPF3C88 E	Number of variable length records &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF36F7 E	Message queue QSYSOPR is allocated to another job.
CPF7304 E	File &1 in &2 not 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.
CPF9807 E	One or more libraries in library list deleted.

CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9812 E	File &1 in library &2 not found.
CPF9814 E	Device &1 not found.
CPF9815 E	Member &5 file &2 in library &3 not found.
CPF9820 E	Not authorized to use library &1.
CPF9821 E	Not authorized to program &1 in library &2.
CPF9822 E	Not authorized to file &1 in library &2.
CPF9825 E	Not authorized to device &1.
CPF9830 E	Cannot assign library &1.
CPF9831 E	Cannot assign device &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R2

Top | Object APIs | APIs by category

Convert Type (QLICVTTP) API

Required Parameter Group:

1	Conversion	Input	Char(10)
2	Symbolic object type	I/O	Char(10)
3	Hexadecimal object type	I/O	Char(2)
4	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Convert Type (QLICVTTP) API lets you convert an object type from the external symbolic format to the internal hexadecimal format and vice versa.

You can use the QLICVTTP API to:

- Convert a specific symbolic object type to an equivalent hexadecimal object type
- Convert a specific hexadecimal object type to an equivalent symbolic object type

Required Parameter Group

Conversion

INPUT; CHAR(10)

The type of conversion to perform.

**HEXTOSYM* An object type in hexadecimal form is converted to an equivalent symbolic object type.

**SYMTOHEX* A symbolic object type is converted to an equivalent hexadecimal form.

Symbolic object type

I/O; CHAR(10)

The external symbolic name given to an object type. An asterisk (*) precedes the object type. The value of the conversion parameter specified determines if this is an input or output field. For a complete list of the available object types, see the Control Language (CL) information in the iSeries Information Center.

Hexadecimal object type

I/O; CHAR(2)

The MI representation of an external object type. This field is in hexadecimal form. The value of the conversion specified determines if this is an input or output field.

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
CPF2101 E	Object type *&1 not valid.
CPF2102 E	Object type and subtype code &1 not valid.
CPF219C E	Conversion value &1 not valid.
CPF219D E	Object type &1 not valid external object type.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R2

Top | Object APIs | APIs by category

List Objects (QUSLOBJ) API

Required Parameter Group:

1	Qualified user space object	Input	Char(20)
2	Format name	Input	Char(8)
3	Object and library name	Input	Char(20)
4	Object type	Input	Char(10)

Optional Parameter Group 1:

5	Error Code	I/O	Char(*)

Optional Parameter Group 2:

6	Authority control	Input	Char(*)
7	Selection control	Input	Char(*)

»Optional Parameter Group 3:

8	Auxiliary storage pool (ASP) control	Input	Char(*)«

Default Public Authority: *USE

Threadsafe: Yes

The List Objects (QUSLOBJ) API lets you generate a list of object names and descriptive information based on specified selection parameters. The QUSLOBJ API places the list in the specified user space. The generated list replaces any existing list in the user space.

You can use the QUSLOBJ API to:

- List objects in a library
- List objects of only one type
- Write an application program to move programs from the QRPLOBJ library»(or the QRPLxxxxx library where 'xxxxx' is the number of a primary auxiliary storage pool)«back to where they were originally located
- Provide backup analysis based on when the object was last saved or last updated
- Provide source member and object analysis from source member information to verify that the current source was used to create the specified object

The QUSLOBJ API returns information in several formats. All formats except OBJL0100 include an information status field that describes the completeness and validity of the information. Be sure to check the information status field before using any other information returned.

Authorities and Locks

If the user is authorized to the library, some object information is always returned for the objects meeting the search criteria identified in the required parameter group. To return any detailed object information in format OBJL0200 and above, the user must be authorized to the objects. The information status field in format OBJL0200 is set to 'A' when the user is not authorized to the objects.

Auxiliary Storage Pool (ASP) Device Authority

*EXECUTE when a specific auxiliary storage pool (ASP) device name is specified for the auxiliary storage pool (ASP) control parameter.

Object Authority

To return detailed object information, any authority except *EXCLUDE is needed when optional parameter group 2 is not specified.

Object Library Authority

*EXECUTE when optional parameter group 2 is not specified.

User Space Authority

*CHANGE

User Space Library Authority

*EXECUTE

User Space Lock

*EXCLRD

Required Parameter Group

Qualified user space object

INPUT; CHAR(20)

The name of the *USRSPC object that is to receive the generated list. The first 10 characters contain the user space object name, and the second 10 characters contain the name of the library where the user space is located. The special values supported for the library name are *LIBL and *CURLIB.

Format name

INPUT; CHAR(8)

The format of the information returned on each object that is requested. You must use one of the following format names:

OBJL0100 Object names (fastest)

OBJL0200 Text description and extended attribute

OBJL0300 Basic object information

OBJL0400 Creation information

OBJL0500 Save and restore information; journal information

OBJL0600 Usage information

OBJL0700 All object information (slowest)

For details about the formats, see Format of the Generated Lists. For performance reasons, you should choose the format that returns only as much information as you need. The higher the number of the format name, the more information is returned and the more time it takes to process.

Object and library name

INPUT; CHAR(20)

The object and library names to place in the *USRSPC object. The first 10 characters contain the object name, which may be a simple name, a generic name, or the special values of *ALL, *ALLUSR, or *IBM. If *ALLUSR or *IBM is used, the library name must be *LIBL or QSYS and the object type parameter must be *LIB.

1. When *ALLUSR is specified with a library name of *LIBL and an object type parameter of *LIB, a list of all user libraries in the thread's library name space is returned. When *LIBL is specified, the auxiliary storage pool (ASP) device name must be an asterisk (*) if the auxiliary storage pool (ASP) control parameter is specified. Refer to *ALLUSR in the description of the second 10 characters of this parameter for a definition of user libraries.

2. When *ALLUSR is specified with a library name of QSYS and an object type parameter of *LIB, a list of all user libraries in the auxiliary storage pools defined by the auxiliary storage pool (ASP) control parameter is returned. Refer to *ALLUSR in the description of the second 10 characters of this parameter for a definition of user libraries.

3. When *IBM is specified with a library name of *LIBL and an object type of *LIB, a list of libraries in the thread's library name space that are saved or restored on the Save Library (SAVLIB) or Restore Library (RSTLIB) CL command with LIB(*IBM) is returned. When *LIBL is specified, the auxiliary storage pool (ASP) device name must be an asterisk (*) if the auxiliary storage pool (ASP) control parameter is specified.

4. When *IBM is specified with a library name of QSYS and an object type of *LIB, a list of libraries in the auxiliary storage pools specified by the auxiliary storage pool (ASP) control parameter that are saved or restored on the Save Library (SAVLIB) or Restore Library (RSTLIB) CL command with LIB(*IBM) is returned.

Library name errors are reported with escape messages when a single library is specified. When searching a set of libraries (library specified as *ALL, *ALLUSR, *LIBL, or *USRLIBL or auxiliary storage pool (ASP) device name specified as *ALLAVL), library errors are reported with diagnostic messages and processing continues. Library authority error messages are not sent when searching a set of libraries. Escape messages are not sent for object name errors. To determine if errors occurred on the object, use the number of list entries field returned in the generic header and the information status field in format OBJL0200.

The second 10 characters identify the name of the library or libraries to search for the specified objects. The following special values are allowed:

**ALL* All libraries in the auxiliary storage pools defined by the auxiliary storage pool (ASP) control parameter are searched.

**ALLUSR* All user libraries in the auxiliary storage pools (ASPs) defined by the auxiliary storage pool (ASP) control parameter are searched. User libraries are all libraries with names that do not begin with the letter Q except for the following:

#CGULIB	#RPGLIB
#COBLIB	#SDALIB
#DFULIB	#SEULIB
#DSULIB	

Although the following libraries with names that begin with the letter Q are provided by IBM, they typically contain user data that changes frequently. Therefore, these libraries are also considered user libraries:

QDSNX	QSYS2xxxxx	QUSROND
QGPL	QS36F	QUSRPOSGS
QGPL38	QUSER38	QUSRPOSSA
QMPGDATA	QUSRADSM	QUSRPYMSVR
QMQMDATA	QUSRBRM	QUSRRDARS
QMQMPROC	QUSRDIRCL	QUSRSYS
QPFRDATA	QUSRDIRDB	QUSRVI
QRCL	QUSRIJS	QUSRVxRxMx
QRCLxxxxx	QUSRINFSKR	
QSYS2	QUSRNOTES	

Notes:

1. 'xxxxx' is the number of a primary auxiliary storage pool (ASP).
2. A different library name, of the form QUSRVxRxMx, can be created by the user for each release that IBM supports. VxRxMx is the version, release, and modification level of the library.

**CURLIB* The thread's current library is searched. When this value is used, the auxiliary storage pool (ASP) device name in the auxiliary storage pool (ASP) control parameter must be an asterisk (*), if specified.

**LIBL* All libraries in the thread's library list are searched. When this value is used, the auxiliary storage pool (ASP) device name in the auxiliary storage pool (ASP) control parameter must be an asterisk (*), if specified.

**USRLIBL* All libraries in the user portion of the thread's library list are searched. When this value is used, the auxiliary storage pool (ASP) device name in the auxiliary storage pool (ASP) control parameter must be an asterisk (*), if specified.

Object type

INPUT; CHAR(10)

The types of objects to search for. You may either enter a specific object type, or a special value of *ALL. For a complete list of the available object types, see the Control Language information.

Optional Parameter Group 1

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

Optional Parameter Group 2

Authority control

INPUT; CHAR(*)

This parameter is used to specify the authority check that should be done for objects and libraries. Detailed object information will only be returned for an object when the user has the specified authority to the object. If this parameter is omitted, the following occurs:

- *EXECUTE authority is checked for on the libraries
- Object name and object type information are always returned for each object. *ANY authority (any authority other than *EXCLUDE) is checked for on the objects to return any detailed information about the objects.
- Call level 0 is used

This parameter is useful to select objects to which a user is authorized. To accomplish this, specify a select or omit status value in the selection control parameter. The object name information in format OBJL0100 is always returned for objects meeting the search criteria identified in the required parameter group. (This assumes the thread has the required authority to the library.) The information status field is set to an A when the thread does not have the object authority specified.

The following example shows what you would specify to obtain a subset of all objects that you have object management authority to.

The authority control parameter would contain:

Length of authority control format: 48

Call level: 1

Displacement to object authorities: 28

Number of object authorities: 1

Displacement to library authorities: 38

Number of library authorities: 1

Object authorities: '*OBJMGT   '

Library authorities: '*USE   '

The selection control parameter would contain:

Length of selection control format: 21

Select or omit status value: 1

Displacement to statuses: 20

Number of statuses: 1

Statuses: 'A'

Because the program that calls the QUSLOBJ API adopts authority, the authority check should be done at the call level previous to the current level (thus call level 1). With call level 1, the list would not include any objects for which you have adopted authority by the current program.

The select or omit status value of 1 indicates that the returned list will omit the objects you do not have object management authority to. This authority is specified in the object authorities field.

The format of this parameter is described in Authority Control Format.

Selection control

INPUT; CHAR(*)

The criteria used to select or filter objects from the list based on specified information status values.

This parameter is useful to reduce the total number of objects returned in the list. The list of objects can be generated with only the specific status that you are interested in. For example, this might be all damaged objects or all objects that the caller of the API is not authorized to. The list of objects also can be generated with all objects except objects of a specific status.

The following example shows what you would specify to select all damaged objects:

Length of selection control format: 22

Select or omit status value: 0

Displacement to statuses: 20

Number of statuses: 2

Statuses: DP

The format of this parameter is described in Selection Control Format.

Optional Parameter Group 3

Auxiliary storage pool (ASP) control

Input; CHAR(*)

The information used to define the auxiliary storage pool (ASP) to search. See Auxiliary Storage Pool (ASP) Control Format for details.

Authority Control Format

The following shows the format of the authority control parameter. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of authority control format
4	4	BINARY(4)	Call level
8	8	BINARY(4)	Displacement to object authorities

12	C	BINARY(4)	Number of object authorities
16	10	BINARY(4)	Displacement to library authorities
20	14	BINARY(4)	Number of library authorities
24	18	BINARY(4)	Reserved
		ARRAY(*) of CHAR(10)	Object authorities
		ARRAY(*) of CHAR(10)	Library authorities

Selection Control Format

The following shows the format of the selection control parameter. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Length of selection control format
4	4	BINARY(4)	Select or omit status value
8	8	BINARY(4)	Displacement to statuses
12	C	BINARY(4)	Number of statuses
16	10	BINARY(4)	Reserved
		ARRAY(*) of CHAR(1)	Statuses

Auxiliary Storage Pool (ASP) Control Format

The following shows the format of the auxiliary storage pool (ASP) control parameter. This parameter is used to define the auxiliary storage pools (ASPs) to search. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Length of auxiliary storage pool (ASP) control format
4	4	CHAR(10)	Auxiliary storage pool (ASP) device name
14	E	CHAR(10)	Auxiliary storage pool (ASP) search type

Field Descriptions

Auxiliary storage pool (ASP) device name. The name of an auxiliary storage pool (ASP) device in which storage is allocated for the library containing the object. The ASP device must have a status of 'Available'. This field must be an asterisk (*) if optional parameter group 3 is specified when *CURLIB, *LIBL, or *USRLIBL is specified as the library name in the object and library name parameter. If optional parameter group 3 is omitted in cases where it is valid for the ASP device name to have a value other than an asterisk (*), the thread's library name space will be used. One of the following special values may be specified:

*	The ASPs in the thread's library name space.
**SYSBAS*	The system ASP (ASP 1) and defined basic user ASPs (ASPs 2-32).
**CURASPGRP*	The ASPs in the current thread's ASP group.
**ALLAVL*	All available ASPs. This includes the system ASP (ASP 1), all defined basic user ASPs (ASPs 2-32), and all available primary and secondary ASPs (ASPs 33-255 with a status of 'Available').

Auxiliary storage pool (ASP) search type. The type of the search when a specific auxiliary storage pool (ASP) device name is specified for the ASP device name field. This field must be blanks when a special value is specified for the auxiliary storage pool (ASP) device name field. One of the following values may be specified:

**ASP*	Only the single ASP named in the auxiliary storage pool (ASP) device name field will be searched.
**ASPGRP*	All ASPs in the auxiliary storage pool (ASP) group named in the auxiliary storage pool (ASP) device name field will be searched. The device name must be the name of the primary auxiliary storage pool (ASP) in the group.

Call level. The number of call levels to go back in the call stack to do the authority check. If optional parameter group 2 is omitted, a call level of 0 is used.

For example, if the program that calls this API adopts authority, you probably would not want the authority check to use the adopted authority. Therefore, the authority check should be done at the call level previous to the current level. This field should then contain a 1. You can check the authority at various call levels by specifying a number equivalent to the call level. For example, to check the authority at the current call level, specify a 0. To check the authority at the previous call level, specify a 1.

This field must be greater than or equal to 0 and less than the number of programs in the call stack.

Displacement to library authorities. The displacement, in bytes, from the beginning of the authority control format to the list of library authorities. The displacement value must be at least 28, which is past the reserved portion of the format.

Displacement to object authorities. The displacement, in bytes, from the beginning of the authority control format to the list of object authorities. The displacement value must be at least 28, which is past the reserved portion of the format.

Displacement to statuses. The displacement, in bytes, from the beginning of the selection control format to the list of statuses requested. The displacement value must be at least 20, which is past the reserved portion of the format.

Length of authority control format. The total length of the authority control format. The length can be 0

bytes to indicate that no authority control information is provided. Otherwise, the minimum size is 48 bytes, which allows for one object and one library authority. An error is returned if the length specified is less than the minimum and not 0.«

Length of auxiliary storage pool (ASP) control format.»The total length of the auxiliary storage pool (ASP) control format. The length can be 0 bytes to indicate that no auxiliary storage pool (ASP) control information is provided. Otherwise, the length must be 24 bytes. An error is returned if the length specified is not 24 or 0.«

Length of selection control format.»The total length of the selection control format. The length can be 0 bytes to indicate that no selection control information is provided. Otherwise, the minimum size is 21 bytes, which allows for one status value. An error is returned if the length specified is less than the minimum and not 0.«

Library authorities. The authority to check for libraries. The array can contain up to ten 10-character fields. If optional parameter group 2 is omitted, *EXECUTE authority is checked for on the libraries.

The authority values can be specified in any combination. If *ALL, *CHANGE, or *USE is specified with any of the other authority values, the authority checked is the cumulative authority value.

The maximum number of authorities that can be specified is 10. This equals all of the specific object and data authorities that can be listed separately.

The following identifies the type of authority the user has to the library:

**ALL*	All authority
**CHANGE*	Change authority
**USE*	Use authority
**OBJOPR*	Object operational authority
**OBJMGT*	Object management authority
**OBJEXIST*	Object existence authority
**OBJALTER*	Alter authority
**OBJREF*	Reference authority
**READ*	Read authority
**ADD*	Add authority
**UPD*	Update authority
**DLT*	Delete authority
**EXECUTE*	Execute authority

Number of library authorities. The number of authorities specified in the library authorities array. You can specify 1 through 10 authorities.

Number of object authorities. The number of authorities specified in the object authorities array. You can specify 1 through 11 authorities.

Number of statuses. The number of statuses specified in the statuses array. You can specify 1 through 5 statuses.

Object authorities. The authority to check for objects. The array can contain up to eleven 10-character fields. If optional parameter group 2 is omitted, *ANY authority is checked for on the objects.

The authority values can be specified in any combination with the exception of the special value *ANY. This must be specified as the only value. If *ALL, *CHANGE, *USE, or *AUTLMGT is specified with any of the other authority values, the authority checked is the cumulative authority value.

The maximum number of authorities that can be specified is 11, which equals all the specific object and data authorities and *AUTLMGT authority.

The following identifies the type of authority the user has to the object:

**ALL*	All authority
**CHANGE*	Change authority
**USE*	Use authority
**AUTLMGT*	Authorization list management authority. (This value is valid only if the object type is *AUTL. It will be ignored for other object types.)
**OBJOPR*	Object operational authority
**OBJMGT*	Object management authority
**OBJEXIST*	Object existence authority
**OBJALTER*	Alter authority
**OBJREF*	Reference authority
**READ*	Read authority
**ADD*	Add authority
**UPD*	Update authority
**DLT*	Delete authority
**EXECUTE*	Execute authority
**ANY*	Any authority other than *EXCLUDE. (If this value is specified, no other values can be specified.)

Reserved. This field is reserved. It must be set to hexadecimal zeros.

Select or omit status value. An indicator that determines whether objects are selected or omitted from the list based on the statuses specified.

This field is useful in generating a list of objects with a certain information status, such as damaged or partially damaged objects. It can also be used to generate a list of all objects except objects with a certain information status, such as unauthorized objects.

Valid values are:

0	Select on status value
1	Omit on status value

Statuses. The status of objects to select or omit from the list of objects generated. Valid values are all of the possible values listed under the information status field (format OBJL0200). The special value * can be used to select all objects with any information status field. If optional parameter group 2 is omitted, all objects with any information status are selected.

Format of the Generated Lists

The object list consists of:

- A user area
- A generic header
- An input parameter section
- A list data section

For details about the user area and generic header, see User space format for list APIs. For details about the other items, see the following sections. For a detailed description of each field in the information returned, see Field Descriptions.

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API Examples.

Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name
10	A	CHAR(10)	User space library name
20	14	CHAR(8)	Format name
28	1C	CHAR(10)	Object name specified
38	26	CHAR(10)	Object library name specified
48	30	CHAR(10)	Object type specified
58	3A	CHAR(2)	Reserved
60	3C	BINARY(4)	Error code bytes provided
64	40	BINARY(4)	Length of authority control format
68	44	BINARY(4)	Call level
72	48	BINARY(4)	Displacement to object authorities
76	4C	BINARY(4)	Number of object authorities
80	50	BINARY(4)	Displacement to library authorities
84	54	BINARY(4)	Number of library authorities
88	58	BINARY(4)	Length of selection control format
92	5C	BINARY(4)	Select or omit status value
96	60	BINARY(4)	Displacement to statuses
100	64	BINARY(4)	Number of statuses

» 104	68	BINARY(4)	Length of auxiliary storage pool (ASP) control format
108	6C	CHAR(10)	Auxiliary storage pool (ASP) device name
118	76	CHAR(10)	Auxiliary storage pool (ASP) search type «
		ARRAY(*) of CHAR(10)	Object authorities
		ARRAY(*) of CHAR(10)	Library authorities
		ARRAY(*) of CHAR(1)	Statuses

OBJL0100 List Data Section

The following information is returned in the list data section of the OBJL0100 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0	CHAR(10)	Object name used
10	A	CHAR(10)	Object library name used
20	14	CHAR(10)	Object type used

OBJL0200 List Data Section

The following information is returned in the list data section of the OBJL0200 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0		Everything from the OBJL0100 format
30	1E	CHAR(1)	Information status
31	1F	CHAR(10)	Extended object attribute
41	29	CHAR(50)	Text description
91	5B	CHAR(10)	User-defined attribute
101	65	CHAR(7)	Reserved

OBJL0300 List Data Section

The following information is returned in the list data section of the OBJL0300 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0		Everything from the OBJL0200 format
108	6C	BINARY(4)	» Object auxiliary storage pool (ASP) number «
112	70	CHAR(10)	Object owner
122	7A	CHAR(2)	Object domain
124	7C	CHAR(8)	Creation date and time
132	84	CHAR(8)	Change date and time
140	8C	CHAR(10)	Storage
150	96	CHAR(1)	Object compression status
151	97	CHAR(1)	Allow change by program
152	98	CHAR(1)	Changed by program
153	99	CHAR(10)	Object auditing value
163	A3	CHAR(1)	Digitally signed
» 164	A4	CHAR(1)	Digitally signed by system-trusted source
165	A5	CHAR(1)	Digitally signed more than once
166	A6	CHAR(2)	Reserved
168	A8	BINARY(4)	Library auxiliary storage pool (ASP) number «

OBJL0400 List Data Section

The following information is returned in the list data section of the OBJL0400 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0		Everything from the OBJL0300 format
172	AC	CHAR(10)	Source file name
182	B6	CHAR(10)	Source file library name
192	C0	CHAR(10)	Source file member name
202	CA	CHAR(13)	Source file updated date and time
215	D7	CHAR(10)	Creator's user profile
225	E1	CHAR(8)	System where object was created
233	E9	CHAR(9)	System level
242	F2	CHAR(16)	Compiler
258	102	CHAR(8)	Object level
266	10A	CHAR(1)	User changed

267	10B	CHAR(16)	Licensed program
283	11B	CHAR(10)	Program temporary fix (PTF)
293	125	CHAR(10)	Authorized program analysis report (APAR)
303	12F	CHAR(10)	Primary group
313	139	CHAR(11)	Reserved

OBJL0500 List Data Section

The following information is returned in the list data section of the OBJL0500 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0		Everything from the OBJL0400 format
324	144	CHAR(8)	Object saved date and time
332	14C	CHAR(8)	Object restored date and time
340	154	BINARY(4)	Save size
344	158	BINARY(4)	Save size multiplier
348	15C	BINARY(4)	Save sequence number
352	160	CHAR(10)	Save command
362	16A	CHAR(71)	Save volume ID
433	1B1	CHAR(10)	Save device
443	1BB	CHAR(10)	Save file name
453	1C5	CHAR(10)	Save file library name
463	1CF	CHAR(17)	Save label
480	1E0	CHAR(8)	Save active date and time
488	1E8	CHAR(1)	Journal status
489	1E9	CHAR(10)	Journal name
499	1F3	CHAR(10)	Journal library name
509	1FD	CHAR(1)	Journal images
510	1FE	CHAR(1)	Journal entries to be omitted
511	1FF	CHAR(8)	Journal start date and time
519	207	CHAR(13)	Reserved

OBJL0600 List Data Section

The following information is returned in the list data section of the OBJL0600 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		

0	0		Everything from the OBJL0500 format
532	214	CHAR(8)	Last-used date and time
540	21C	CHAR(8)	Reset date and time
548	224	BINARY(4)	Days-used count
552	228	CHAR(1)	Usage information updated
» 553	229	CHAR(10)	Object auxiliary storage pool (ASP) device name
563	233	CHAR(10)	Library auxiliary storage pool (ASP) device name
573	23D	CHAR(3)	Reserved«

OBJL0700 List Data Section

The following information is returned in the list data section of the OBJL0700 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset			
Dec	**Hex**	**Type**	**Field**
0	0		Everything from the OBJL0600 format
576	240	BINARY(4)	Object size
580	244	BINARY(4)	Object size multiplier
584	248	CHAR(1)	Object overflowed ASP indicator
585	249	CHAR(3)	Reserved

Field Descriptions

Allow change by program. A 1-character variable that is used to return the allow change by program flag. A 1 is returned if the object can be changed with the Change Object Description (QLICOBJD) API. A 0 is returned if the object cannot be changed with the API.

Authorized program analysis report (APAR). The identifier of the authorized program analysis report (APAR) that caused this object to be replaced. The field is blank if the object did not change because of an APAR.

»**Auxiliary storage pool (ASP) device name.** The name of the auxiliary storage pool (ASP) device to be searched for the library, as specified in the call to the API.

Auxiliary storage pool (ASP) search type. The type of the auxiliary storage pool (ASP) search, as specified in the call to the API.«

Call level. The number of call levels to go back in the call stack to do the authority check. If optional parameter group 2 is omitted, a call level of 0 is used.

For example, if the program that calls this API adopts authority, you would probably not want the authority check to use the adopted authority. Therefore, the authority check should be done at the call level previous

to the current level. This field should then contain a 1. You can check the authority at various call levels by specifying a number equivalent to the call level. For example, to check the authority at the current call level, specify a 0. To check the authority at the previous call level, specify a 1.

This field must be greater than or equal to 0 and less than the number of programs in the call stack.

Changed by program. A 1-character variable that is used to return the changed by program flag. A 1 is returned if the object has been changed with the QLICOBJD API. A 0 is returned if the object has not been changed by the API.

Change date and time. The time at which the object was last changed, in system time-stamp format.

Compiler. The licensed program identifier, version number, release level, and modification level of the compiler. The field has a pppppppVvvRrrMmm format where:

ppppppp	The licensed program identifier.
Vvv	The character V is followed by a 2-character version number.
Rrr	The character R is followed by a 2-character release level.
Mmm	The character M is followed by a 2-character modification level.

The field is blank if you do not compile the program.

Creation date and time. The time at which the object was created, in system time-stamp format. See the Convert Date and Time Format (QWCCVTDT) API for more information about using this time-stamp format.

Creator's user profile. The name of the user that created the object.

Days-used count. The number of days the object was used. If the object does not have a last-used date, the count is 0.

Digitally signed. A 1-character variable that indicates whether the object has an OS/400 digital signature.

0 The object does not have an OS/400 digital signature.

1 The object has an OS/400 digital signature.

Digitally signed by system-trusted source. A 1-character variable indicates whether the object is signed by a source that is trusted by the system.

0 None of the object signatures came from a source that is trusted by the system.

1 The object is signed by a source that is trusted by the system. If the object has multiple signatures, at least one of the signatures came from a source that is trusted by the system.

Digitally signed more than once. A 1-character variable that indicates whether the object has more than one OS/400 digital signature.

0 The object has only one OS/400 digital signature or does not have an OS/400 digital signature. Refer to the digitally signed variable to determine whether the object has an OS/400 digital signature.

1 The object has more than one OS/400 digital signature. Refer to the digitally signed by system-trusted source variable to determine whether the object has an OS/400 digital signature from a source trusted by the system.

Displacement to library authorities. The displacement, in bytes, from the beginning of this input parameter structure to the list of library authorities, as specified in the call to the API.

Displacement to object authorities. The displacement, in bytes, from the beginning of this input parameter structure to the list of object authorities, as specified in the call to the API.

Displacement to statuses. The displacement, in bytes, from the beginning of this input parameter structure to the list of statuses requested, as specified in the call to the API.

Error code bytes provided. The length of the area that the calling application provides for the error code, in bytes.

Extended object attribute. The extended attribute of the object, such as a program or file type. Extended attributes further describe the object. For example, an object type of *PGM may have a value of RPG (RPG program) or CLP (CL program), and an object type of *FILE may have a value of PF (physical file), LF (logical file), DSPF (display file), SAVF (save file), and so on.

Format name. The format of the returned output.

Information status. Whether or not the QUSLOBJ API returns the requested information for this object. Possible values are:

blank	The requested information is returned. No errors occurred.
A	No information is returned. The thread that called this API needs either the authority specified in the object authorities field or *ANY authority (the default) to the object.
D	The requested information is returned. However, the object is damaged and should be re-created as soon as possible.
L	No information is returned because the object is locked.
P	The requested information is returned. However, the object is partially damaged. In most instances, to recover from partial object damage, you delete the damaged object and either restore a saved copy or create the object again. For some damaged objects, special recovery procedures are possible. Refer to the Backup and Recovery book for more information on damaged objects.

If two or more conditions occur that include no authorization (A) to the object, the status is set to A. If the object is damaged (D) and locked (L), or if the object is partially damaged (P) and locked, the status is set to L.

If the value of this field is A or L, your application should not use the other fields for the object. Only the object name, library, and type fields contain accurate data.

Journal entries to be omitted. The journal entries to be omitted. The field is 1 if *open* and *close* operations do not generate *open* and *close* journal entries. The field is 0 if no entries are omitted. This field is blank if the object has never been journaled.

Journal images. The type of images that are written to the journal receiver for updates to the object. The field is 0 if only *after* images are generated for changes to the object. The field is 1 if both *before* and *after* images are generated for changes to the object. This field is blank if the object has never been journaled.

Journal library name. The name of the library containing the journal. This field is blank if the object has never been journaled.

Journal name. The name of the current or last journal. This field is blank if the object has never been journaled.

Journal start date and time. The time at which journaling for the object was last started, in system time-stamp format. This field contains hexadecimal zeros if the object has never been journaled.

Journal status. The 1-character variable that returns the current journaling status of an object. The value is 1 if the object currently is being journaled; the value is 0 if the object currently is not being journaled.

Last-used date and time. The date and time at which the object was last used, in system time-stamp format. If the object has no last-used date, the field contains hexadecimal zeros.

Length of authority control format. The total length of the authority control format, »as specified in the call to the API.«

»**Length of auxiliary storage pool (ASP) control format.** The total length of the auxiliary storage pool (ASP) control format, as specified in the call to the API.«

Length of selection control format. The total length of the selection control format, »as specified in the call to the API.«

Library authorities. The authority to check for libraries. The array can contain up to ten 10-character fields. If optional parameter group 2 is omitted, *EXECUTE authority is checked for on the libraries.

The authority values can be specified in any combination. If *ALL, *CHANGE, or *USE is specified with any of the other authority values, the authority checked is the cumulative authority value.

The maximum number of authorities that can be specified is 11. This equals all of the specific object and data authorities that can be listed separately.

The following identifies the type of authority the user has to the library:

**ALL*	All authority
**CHANGE*	Change authority
**USE*	Use authority
**OBJOPR*	Object operational authority
**OBJMGT*	Object management authority
**OBJEXIST*	Object existence authority
**OBJALTER*	Alter authority
**OBJREF*	Reference authority
**READ*	Read authority
**ADD*	Add authority
**UPD*	Update authority
**DLT*	Delete authority
**EXECUTE*	Execute authority

»**Library auxiliary storage pool (ASP) device name.** The name of the auxiliary storage pool (ASP)

device where storage is allocated for the library containing the object. The following special value may be returned:

**SYSBAS* System ASP (ASP 1) or basic user ASPs (ASPs 2-32)

Library auxiliary storage pool (ASP) number. The number of the auxiliary storage pool (ASP) where storage is allocated for the library containing the object. Valid values are:

1 System ASP

2-32 Basic user ASP

33-255 Primary or secondary ASP

Licensed program. The name, release level, and modification level of the licensed program if the retrieved object is part of a licensed program. The 7-character name starts in character position 1, the version number starts in position 8, the release level starts in position 11, and the modification level starts in position 14. The field is blank if the retrieved object is not a part of a licensed program.

Number of library authorities. The number of authorities specified in the library authorities array. You can specify 1 through 10 authorities.

Number of object authorities. The number of authorities specified in the object authorities array. You can specify 1 through 11 authorities.

Number of statuses. The number of statuses specified in the statuses array. You can specify 1 through 5 statuses.

Object auditing value. A 10-character variable that is used to return the type of auditing for an object. The valid values are:

**NONE* No auditing occurs for this object when it is read or changed regardless of the user who is accessing the object.

**USRPRF* Audit this object only if the user accessing the object is being audited. The user profile for the thread is tested to determine if auditing should be done for this object. The user profile can specify if only change access is audited or if both read and change accesses are audited for this object.

**CHANGE* Audit all change access to this object by all users on the system.

**ALL* Audit all access to this object by all users on the system. All access is defined as a read or change operation.

Object authorities. The authority to check for objects. The array can contain up to eleven 10-character fields. If optional parameter group 2 is omitted, *ANY authority is checked for on the objects.

The authority values can be specified in any combination with the exception of the special value *ANY. This must be specified as the only value. If *ALL, *CHANGE, *USE, or *AUTLMGT is specified with any of the other authority values, the authority checked is the cumulative authority value.

The maximum number of authorities that can be specified is 11, which equals all the specific object and data authorities and *AUTLMGT authority.

The following identifies the type of authority the user has to the object:

**ALL* All authority

**CHANGE*	Change authority
**USE*	Use authority
**AUTLMGT*	Authorization list management authority. (This value is valid only if the object type is *AUTL. It will be ignored for other object types.)
**OBJOPR*	Object operational authority
**OBJMGT*	Object management authority
**OBJEXIST*	Object existence authority
**OBJALTER*	Alter authority
**OBJREF*	Reference authority
**READ*	Read authority
**ADD*	Add authority
**UPD*	Update authority
**DLT*	Delete authority
**EXECUTE*	Execute authority
**ANY*	Any authority other than *EXCLUDE. (If this value is specified, no other values can be specified.)

»**Object auxiliary storage pool (ASP) device name** The name of the auxiliary storage pool (ASP) device where storage is allocated for the object. The following special value may be returned:

**SYSBAS*	System ASP (ASP 1) or defined basic user ASPs (ASPs 2-32)

Object auxiliary storage pool (ASP) number. The number of the auxiliary storage pool (ASP) where storage is allocated for the object. Valid values are:

1	System ASP
2-32	Basic user ASP
33-255	Primary or secondary ASP«

Object compression status. Whether the object is compressed or decompressed. The status is returned in a 1-character variable with one of these values:

Y	Compressed.
N	Permanently decompressed and compressible.
X	Permanently decompressed and *not* compressible.
T	Temporarily decompressed.
F	Saved with storage freed; compression status cannot be determined.

Temporarily decompressed objects exist in both decompressed and compressed form. *Permanently*

decompressed objects exist in decompressed form only. The system handles some decompression automatically, depending on the type of object, the operation performed on it, and its frequency of use. For an overview of object compression and decompression, see the CL Programming book. For details about how to explicitly compress and decompress objects, see the entries for these commands in the Control Language (CL) information: Compress Object (CPROBJ), Decompress Object (DCPOBJ), and Reclaim Temporary Storage (RCLTMPSTG).

Object domain. The domain that contains the object. The value is *U if the object is in the user domain, or *S if the object is in the system domain.

Object level. The object control level for the created object.

Object library name specified. The name of the object library, as specified in the call to the API.

Object library name used. The name of the library containing the object.

Object name specified. The name of the object, as specified in the call to the API.

Object name used. The name of the object.

Object overflowed ASP indicator. The 1-character variable that returns the object overflowed auxiliary storage pool (ASP) indicator. The value is 1 if the object overflowed the ASP in which it resides; the value is 0 if the object has not overflowed the ASP. For objects in the system ASP (ASP 1) or in a primary or secondary ASP (ASPs 33-255), a 0 is always returned because it is not possible for an object that resides in the system ASP or in a primary or secondary ASP to overflow its ASP.

Object owner. The name of the object owner's user profile.

Object restored date and time. The time at which the object was restored, in system time-stamp format. If the object has never been restored, the field contains hexadecimal zeros.

Object saved date and time. The time at which the object was saved, in system time-stamp format. If the object has never been saved, the field contains hexadecimal zeros.

Object size. The size of the object in units of the size multiplier. The object size is equal to or smaller than the object size multiplied by the object size multiplier.

Object size multiplier. The value to multiply the object size by to get the true size. The value is 1 if the object is smaller than 1 000 000 000 bytes, and 1024 if it is larger.

Object type specified. The type of the object, as specified in the call to the API.

Object type used. The type of the object. For a list of all the available object types, see the Control Language information.

Primary group. The name of the user who is the primary group for the object. If no primary group exists for the object, this field contains a value of *NONE.

Program temporary fix (PTF). The number of the program temporary fix (PTF) number that caused this object to be replaced. This field is blank if the object was not changed because of a PTF.

Reserved. An unused field. It contains hexadecimal zeros.

Reset date and time. The date the days-used count was last reset to zero, in system time-stamp format. If the days-used count has never been reset, the field contains hexadecimal zeros.

Save active date and time. The date and time the object was last saved when the SAVACT(*LIB,

*SYSDFN, or *YES) save operation was specified, in system time-stamp format. This parameter is found on the Save Library (SAVLIB), Save Object (SAVOBJ), Save Changed Object (SAVCHGOBJ), and Save Document Library Object (SAVDLO) CL commands. If the object has never been saved or if SAVACT(*NO) was specified on the last save operation for the object, the field contains hexadecimal zeros.

Save command. The command used to save the object. The field is blank if the object was not saved.

Save device. The type of device to which the object was last saved. The field is *SAVF if the last save operation was to a save file. The field is *DKT if the last save operation was to diskette. The field is *TAP if the last save operation was to tape. The field is *OPT if the last save operation was to optical. The field is blank if the object was not saved.

Save file library name. The name of the library that contains the save file if the object was saved to a save file. The field is blank if the object was not saved to a save file.

Save file name. The name of the save file if the object was saved to a save file. The field is blank if the object was not saved to a save file.

Save label. The file label used when the object was saved. The variable is blank if the object was not saved to tape, diskette, or optical. The value of the variable corresponds to the value specified for the LABEL of OPTFILE parameter on the command used to save the object.

Save sequence number. The tape sequence number assigned when the object was saved on tape. If the object was not saved to tape, the field contains zeros.

Save size. The size of the object in bytes of storage at the time of the last save operation. The save size is equal to or smaller than the save size multiplied by the save size multiplier. The field contains zeros if the object was not saved.

Save size multiplier. The value to multiply the save size by to get the true size. The value is 1 if the save size is smaller than 1 000 000 000 bytes, and 1024 if it is larger.

Save volume ID. The tape, diskette, or optical volumes that are used for saving the object. The variable returns a maximum of 10 six-character volumes. The volume IDs begin in character positions 1, 8, 15, 22, 29, 36, 43, 50, 57, and 64. Each volume ID entry is separated by a single character. If the object was saved in parallel format, the separator character contains a 2 before the first volume in the second media file, a 3 before the third media file, and so on, up to a 0 before the tenth media file. Otherwise, the separator characters are blank. If more than 10 volumes are used and the object was saved in serial format, 1 is returned in the 71st character of the variable. If the object was saved in parallel format, a 2 is returned in the 71st character of the variable. Otherwise, the 71st character is blank. The field is blank if the object was last saved to a save file or if it was never saved.

Select or omit status value. An indicator that determines whether objects are selected or omitted from the list based on the statuses specified.

This field is useful in generating a list of objects with a certain information status, such as damaged or partially damaged objects. It can also be used to generate a list of all objects except objects with a certain information status, such as unauthorized objects.

0 Select on status value

1 Omit on status value

Source file library name. The name of the library that contains the source file used to create the object. The field is blank if no source file created the object.

Source file member name. The name of the member in the source file. The field is blank if no source file created the object.

Source file name. The name of the source file used to create the object. The field is blank if no source file created the object.

Source file updated date and time. The date and time the member in the source file was last updated. This field is in the CYYMMDDHHMMSS format where:

C	Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
YY	Year
MM	Month
DD	Day
HH	Hour
MM	Minute
SS	Second

The field is blank if no source file created the object.

Statuses. The status of objects to select or omit from the list of objects generated. Valid values are all of the possible values listed under the information status field (format OBJL0200). The special value * can be used to select all objects with any information status field. If optional parameter group 2 is omitted, all objects with any information status are selected.

Storage. The storage status of the object data. *FREE indicates the object data is freed and the object is suspended. *KEEP indicates the object data is not freed and the object is not suspended.

System level. The level of the operating system when the object was created. The field has a VvvRrrMmm format where:

Vvv	The character V is followed by a 2-character version number.
Rrr	The character R is followed by a 2-character release level.
Mmm	The character M is followed by a 2-character modification level.

System where object was created. The name of the system on which the object was created.

Text description. The text description of the object. The field is blank if no text description is specified.

Usage information updated. Whether the object usage information is updated for this object type. The indicator is returned as Y (Yes) or N (No).

User changed. Whether the user program was changed. A character 1 is returned if the user changed the object. If the object was not changed by the user, the field is character 0.

User-defined attribute. Further defines an object type. This field is set by the user while using the QLICOBJD API.

User space library name. The library containing the user space, as specified in the call to the API.

User space name. The name of the user space.

Error Messages

Message ID	Error Message Text
» CPFB8ED E	Device description &1 not correct for operation.«
CPF21A7 E	Authority value &1 not valid.
CPF21A8 E	Must specify *ANY as only authority value.
CPF21A9 E	Select or omit value &1 not valid.
CPF21AA E	Number of statuses must be between 1 and 5.
CPF21AB E	Status value &1 not valid.
CPF21AC E	Length or displacement value &1 not valid.
» CPF2173 E	Value for ASPDEV not valid with special value for library.«
» CPF218C E	&1 not a primary or secondary ASP.«
» CPF218D E	&1 not a primary ASP when *ASPGRP specified.«
CPF22F7 E	Number of authorities must be between 1 and &1.
CPF22F9 E	Call level &1 not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CAA E	List is too large for user space &1.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C20 E	Error found by program &1. CPD3C21 D Format name &1 is not valid. CPD3C31 D Object type &1 is not valid.
CPF3C21 E	Format name &1 is not valid.
» CPF3C3B E	Value for parameter &2 for API &1 not valid.«
CPF3C31 E	Object type &1 is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
» CPF980B E	Object &1 in library &2 not available.«
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.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
» CPF9814 E	Device &1 not found.«
CPF9820 E	Not authorized to use library &1.
» CPF9825 E	Not authorized to device &1.«
CPF9830 E	Cannot assign library &1.
» CPF9833 E	*CURASPGRP or *ASPGRPPRI specified and thread has no ASP group.«
CPF9838 E	User profile storage limit exceeded.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V1R3

Top | Object APIs | APIs by category

Materialize Context (QusMaterializeContext) API

Required Parameter Group:

1	Receiver	I/O	PTR(SPP)
2	Context	Input	PTR(SYP)
3	Materialize options	Input	Char(*)

Default Public Authority: *USE

Service Program: QUSMIAPI

Threadsafe: No

The term context in this API is synonymous with the OS/400 term library.

The Materialize Context (QusMaterializeContext) API returns either the type and subtype of the object or system pointers, based on what you specify for the materialize options parameter. The API returns the information for all or for a selected set of objects that are contained by the context. This information is returned to a receiver variable. If the context is null, the machine context (the QSYS library) is returned.

This API provides the function of the MATCTX MI instruction on all security levels of OS/400. See the MATCTX instruction in the iSeries Machine Interface Instructions for the documentation of this API.

Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
MCHxxxx E	See the iSeries Machine Interface Instructions for exact MCH messages that could be signaled.

API introduced: V3R7

Move Folder to ASP (QHSMMOVF) API

Required Parameter Group:

1	Folder name	Input	Char(12)
2	Target auxiliary storage pool (ASP)	Input	Binary(4)
3	Error code	I/O	Char(*)

Default Public Authority: *USE

The Move Folder to ASP (QHSMMOVF) API moves a root folder and its contents from its existing auxiliary storage pool (ASP) to the specified target ASP through a save and restore process. The API, however, will retain private authorities to the objects that would normally be lost with a save and restore operation.

Restrictions

The Move Folder to ASP (QHSMMOVF) API has the following restrictions:

- The folder must be a root folder and will be moved as such.
- The folder and its contents must not be in use by other jobs.
- Folders that were restored using the Restore Licensed Program (RSTLICPGM) should not be moved.
- After the root folder has been moved, the following parameters are changed:
  - The date last used will be set to blank.
  - The change date and time will be set to the current date and time.
  - The days used count will be set to zero.
  - The days used count reset will be set to blank.
  - The save date/time and restore date/time will be updated.
- The target ASP must have enough space for the folder and its objects in order for the API to perform the move action.
- The target ASP must be either the system ASP, a library-type ASP, or an empty ASP.
- The user must be enrolled in the system distribution directory.
- Access codes are the responsibility of the user and will be lost.
- A root folder that contains documents checked out or saved with STG(*FREE) will not be moved.
- A root folder that contains more than 99 subfolders will not be moved.

Authorities and Locks

See Lock Conditions When Saving and Restoring Objects in the Backup and Recovery book for detailed information on locks that are applied to objects during save operations.

To prevent access to other jobs, the folder is renamed to QHSMFLR.xxx, where xxx is a numeric increment to allow multiple concurrent move operations.

The user profile must have *OBJALTER authority to the folder and its objects if it does not have *ALLOBJ authority.

See Authority Required for Objects Used by Commands in the iSeries Security Reference book for detailed information on object authorities required when you save objects.

Required Parameter Group

Folder name

INPUT; CHAR(12)

The name of the folder to be moved to the specified auxiliary storage pool (ASP).

Target auxiliary storage pool (ASP)

INPUT; BINARY(4)

The number of the auxiliary storage pool (ASP) to which the folder is to be moved. You can use one of the following values for the ASP number:

1-16 The number that was assigned to the ASP at creation time. ASP 1 is the system ASP. This ASP must be different from the ASP where the folder exists.

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
CPF2115 E	Object &1 in &2 type *&3 damaged.
CPF3C90 E	Literal value cannot be changed.
CPF8A00 E	All CPF8Axx messages could be signaled. xx is from 01 to FF.
CPF9000 E	All CPF90xx messages could be signaled. xx is from 01 to FF.

CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFB78B E	Errors encountered while moving or migrating folder &1.
CPFB781 E	Not authorized to folder &1.
CPFB783 E	Folder &1 cannot be moved or migrated into an object-based ASP &2.
CPFB784 E	Target ASP &1 is not valid for the move or migrate operation.
CPFB786 E	Insufficient disk capacity in ASP &1 for specified objects.
CPFB79E E	Auxiliary storage pool &1 does not exist.
CPFB790 E	Move or migrate of folder &1 failed.
CPFB799 E	Unexpected condition with &1 API. Reason & 6.

API Introduced: V4R3

Top | Object API categories | API by category

Move Library to ASP (QHSMMOVL) API

Required Parameter Group:

1	Library name	Input	Char(10)
2	» Target auxiliary storage pool (ASP) number«	Input	Binary(4)
3	Check dependencies	Input	Char(10)
4	Error Code	I/O	Char(*)

»Optional Parameter Group 1:

5	Target auxiliary storage pool (ASP) device name	Input	Char(10)
6	Source auxiliary storage pool (ASP) number	Input	Binary(4)
7	Source auxiliary storage pool (ASP) device name	Input	Char(10)«

Default Public Authority: *USE

Threadsafe: No

The Move Library to ASP (QHSMMOVL) API moves a library and its contents from its existing auxiliary storage pool (ASP) to the specified target ASP through a save and restore process. The API, however, will preserve private authorities to the objects that would normally be lost with a save and restore operation.

Restrictions

The Move Library to ASP (QHSMMOVL) API has the following restrictions:

- When you move a library from the system ASP to a user ASP, libraries and their objects are checked for eligibility.
  - Libraries that begin with 'Q' are considered to be system libraries and will not be moved.

    See the Backup and Recovery book for detailed information on the list of objects that cannot be moved.
  - »If the library cannot be renamed, the library is not allowed to be moved. See the RNMOBJ (Rename Object) Control Language (CL) information in the iSeries Information Center for the restrictions on renaming a library.«
- Libraries that contain journal objects or journaled files are not allowed to be moved.
- Libraries that contain files with database dependencies outside the library are not allowed to be moved.
- Job queue and output queue entries are moved if the queue is not allocated.

- Data queue entries are the responsibility of the user and will be lost.
- »A library cannot be moved if it is in the library list of the current thread.«
- »A library cannot be moved if it is in the library list of any primary thread that is active on the system when the QLIBLCKLVL system value is set to lock libraries in the library list.«
- The QSYSWRK subsystem must be active.
- Program (*PGM) objects in a library will be placed in library QRPLOBJ »(or library QRPLxxxxx if the library is in a primary ASP with an ASP number corresponding to 'xxxxx' (where 'xxxxx' is the ASP number right adjusted and padded on the left with zeros) or in a secondary auxiliary storage pool in the same group as that primary ASP)« and another copy of each *PGM object will be moved with the library to the target ASP.
- After a library has been moved, the following attributes are changed
  - The date last used will be set to blank.
  - The change date and time will be set to the current date and time.
  - The days used count will be set to zero.
  - The date use count reset will be set to blank.
  - The restore date and time will be set to the current date and time.
- The target ASP must have enough space for the library and its objects in order for the API to perform the move operation.
- »The target ASP must be either the system ASP (ASP 1), a library-type or empty basic user ASP (ASPs 2-32), or a primary or secondary ASP (ASPs 33-255).«

Authorities and Locks

»The user profile must have *USE authority to the auxiliary storage pool (ASP) device when a specific ASP device name is specified for the source or target auxiliary storage pool (ASP) device name or when a value greater than 32 is specified for the source or target auxiliary storage pool (ASP) number.«

See Lock Conditions When Saving and Restoring Objects in the Backup and Recovery book for detailed information on locks that are applied to objects during save operations.

To prevent access to other jobs, the library is renamed to QHSMLIBxxx where xxx is a numeric increment to allow multiple concurrent move operations.

The user profile must have *OBJALTER authority to the library if it does not have *ALLOBJ authority.

See Authority Required for Objects Used by Commands in the iSeries Security Reference book for detailed information on object authorities required when you save objects.

Required Parameter Group

Library name

INPUT; CHAR(10)

The name of the library to be moved to the specified auxiliary storage pool (ASP).

»Target auxiliary storage pool (ASP) number«

INPUT; BINARY(4)

The number of the auxiliary storage pool (ASP) to which the library is to be moved. The following values can be used for the ASP number:

»*1-255* The number that was assigned to the ASP at creation time. ASP 1 is the system ASP. ASP numbers 2-32 are basic user ASPs. ASP numbers 33-255 are primary or secondary ASPs. This ASP must be different from the ASP where the library exists.

-1 The ASP value is specified on the target auxiliary storage pool (ASP) device name parameter.«

Check dependencies

INPUT; CHAR(10)

The API checks for object, database and journal dependencies. You can use one of the following values:

**YES* The API will check for object dependencies, database dependencies outside the library, and for journal dependencies. An exception is signaled in case any of the conditions are found.

**NO* » The API bypasses the validation for object and journal dependencies.« This value could be used if the user knows that dependencies do not exist to save on processing time. However, the API will end abnormally during the move operation if dependency conditions are encountered.

**VALIDATE* The API will perform the validation for object, database, and journal dependencies. An information message will be sent to the caller of the API indicating that the library can be moved to the selected ASP. This option is best used for planning purposes to determine which libraries can be moved to a specific ASP.

Error code

I/O; CHAR(*)

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

»Optional Parameter Group 1

Target auxiliary storage pool (ASP) device name

INPUT; CHAR(10)

The name of the auxiliary storage pool (ASP) device from which storage is allocated for the library after the move. The ASP device must have a status of 'Available'. If this parameter is omitted, the target auxiliary storage pool (ASP) number parameter will be used. This parameter must be *TGTASP if specified when the target auxiliary storage pool (ASP) number parameter has a value other than -1. The ASP device must be different from the ASP device where the library exists.

One of the following special values may be specified:

**TGTASP*	The ASP is determined from the target auxiliary storage pool (ASP) number parameter.
**CURGRPPRI*	If the thread has an ASP group, the storage for the library is allocated from the the primary ASP of the group.
**SYSTEM*	The system ASP (ASP 1).

Source auxiliary storage pool (ASP) number

INPUT; BINARY(4)

The number of the auxiliary storage pool (ASP) from which the library is to be moved. If this parameter and the source auxiliary storage pool (ASP) device name parameter are both omitted, the thread's library name space will be searched for the library.

The following values for the ASP number can be used:

1-255	The number that was assigned to the ASP at creation time. ASP 1 is the system ASP. ASP numbers 2-32 are basic user ASPs. ASP numbers 33-255 are primary or secondary ASPs. The primary or secondary ASP must have a status of 'Available'.
-1	The ASP value is specified on the source auxiliary storage pool (ASP) device name parameter.

Source auxiliary storage pool (ASP) device name

INPUT; CHAR(10)

The name of the auxiliary storage pool (ASP) device from which storage is allocated for the library to be moved. The ASP device must have a status of 'Available'. This parameter must be *ASP if specified when the source auxiliary storage pool (ASP) number parameter has a value other than -1. If this parameter and the source auxiliary storage pool (ASP) number parameter are both omitted, the thread's library name space will be searched for the library.

One of the following special values may be specified:

**ASP*	The ASP to be searched is determined from the source auxiliary storage pool (ASP) number parameter.
*	The ASPs in the thread's library name space will be searched for the library.
**CURASPGRP*	The ASPs in the thread's ASP group will be searched for the library.
**SYSBAS*	The system ASP (ASP 1) and defined basic user ASPs (ASPs 2-32) will be searched for the library.«

Error Messages

Message ID	Error Message Text
CPFB78A E	Job queue &1 in library &2 is attached to subsystem &3.
» CPFB78C E	Library &1 not moved or migrated.«
CPFB78D E	Library &1 not moved or migrated.
CPFB78E E	Move or migrate of library &1 failed.
» CPFB78F E	Move or migrate of library &1 failed«
CPFB780 E	Not authorized to library &1.
CPFB782 E	Library &1 cannot be moved or migrated into an object-based ASP &2.
CPFB784 E	Target ASP &1 is not valid for the move or migrate operation.
CPFB785 E	Library &1 contains objects not valid for selected ASP.
CPFB786 E	Insufficient disk capacity in ASP &1 for specified objects.
CPFB787 E	Library &1 has journal dependencies.
CPFB788 E	Library &1 has database dependencies.
CPFB789 E	Output queue &1 in library &2 is attached to writer &3.
CPFB79E E	Auxiliary storage pool &1 does not exist.
CPFB791 E	Cannot access object &1 in library &2 type &3.
CPFB792 E	An error was encountered while moving or migrating library &1.
CPFB793 E	Move or migrate of job queue entries failed.
CPFB794 E	Move of output queue entries failed.
CPFB795 E	» Library &1 cannot be allocated.«
CPFB799 E	Unexpected condition with &1 API. Reason &6.
» CPFB8ED E	Device description &1 not correct for operation.«
CPF2115 E	Object &1 in &2 type *&3 damaged.
» CPF2166 E	Library name &1 not valid.«
» CPF218C E	&1 not a primary or secondary ASP.«
» CPF24B4 E	Severe error while addressing parameter list.«
» CPF3C36 E	Number of parameters, &1, entered for this API was not valid.«
» CPF3C3C E	Value for parameter &1 not valid.«
CPF3C90 E	Literal value cannot be changed.

CPF9810 E	Library &1 not found.
» CPF9814 E	Device &1 not found.«
»CPF9825 E	Not authorized to device &1.«
»CPF9833 E	*CURASPGRP or *CURGRPPRI specified and thread has no ASP group.«
»CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.«

API Introduced: V4R3

Top | Object APIs | APIs by category

Open List of Objects (QGYOLOBJ) 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	Sort information	Input	Char(*)
6	Object and library name	Input	Char(20)
7	Object type	Input	Char(10)
8	Authority control	Input	Char(*)
9	Selection control	Input	Char(*)
10	Number of fields to return	Input	Binary(4)
11	Key of fields to return	Input	Array(*) of Binary(4)
12	Error Code	I/O	Char(*)

Optional Parameter Group 1:

13	Job identification information	Input	Char(*)
14	Format of job identification information	Input	Char(8)

» Optional Parameter Group 2:

15	Auxiliary storage pool (ASP) control	Input	Char(*) «

Default Public Authority: *USE

Threadsafe: No

The Open List of Objects (QGYOLOBJ) API lets you generate a list of object names and descriptive information based on specified selection parameters. The QGYOLOBJ API places the list into a receiver variable. You can access additional records by using the Get List Entries (QGYGTLE) API. On 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 Entry Number in List (QGYFNDE)

Close List (QGYCLST)

You can use the QGYOLOBJ API to:

- Open a list of objects in a library
- Open a list of objects of only one type
- Open a list of objects in another thread's library list

- Write an application program to move programs from the QRPLOBJ library back to their original location
- Provide backup analysis based on when the object was last saved or last updated
- Provide source member and object analysis from source member information to verify that the current source was used to create the specified object

The records returned by QGYOLOBJ include an information status field that describes the completeness and validity of the information. Be sure to check the information status field before using any other information returned.

Note: The QTEMP library and the system portion of the library list could be different between the main job and the server job when the list is being built asynchronously. If this is a problem, then request that the list be built synchronously.

For more information, see the Process Open List APIs within the OS/400 Miscellaneous APIs.

Authorities and Locks

If the user is authorized to the library, some object information is always returned for the objects meeting the search criteria identified in the required parameter group. To return any of the data identified in the key fields, the user must be authorized to the objects. The information status field in the receiver variable is set to 'A' when the user is not authorized to the objects.

Auxiliary Storage Pool (ASP) Device Authority

*EXECUTE when a specific auxiliary storage pool (ASP) device name is specified for the auxiliary storage pool (ASP) control parameter.

Object Authority

Authority specified in the Authority control parameter

Object Library Authority

Authority specified in the Authority control parameter

Job Authority

When optional parameter group 1 is specified, the API must be called from within the thread for which the object list is being retrieved, or the caller of the API must be running under a user profile that is the same as the job user identity of the job for which the object list is being retrieved. Otherwise, the caller of the API must be running under a user profile that has job control (*JOBCTL) special authority.

The **job user identity** is the name of the user profile by which a job is known to other jobs. It is described in more detail in the Work Management book on the V5R1 Supplemental Manuals Web site.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable used to return the number of records requested (given in the number of records to return parameter) of object information. For details about the structure of the receiver variable, see Format of Receiver Variable.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable.

List information

OUTPUT; CHAR(80)

Information about the list of objects that was opened. For a description of the layout of this parameter, see Format of open list information.

Number of records to return

INPUT; BINARY(4)

The number of records in the list to put into the receiver variable after filtering and sorting has been done. Possible values follow:

-1 All records are built synchronously in the list.

0 All records are built asynchronously in the list.

If a positive number of records is specified at least that many are built synchronously (in order to return those immediately to the caller of this API) and the remainder are built asynchronously by a server job.

Sort information

INPUT; CHAR(*)

Information about which fields within the record structure to sort. For the layout of this structure, see Sort Information Format.

Object and library name

INPUT; CHAR(20)

The object and library names to place in the list. The first 10 characters contain the object name, which may be a simple name, a generic name, or the special values *ALL, *ALLUSR, or *IBM. If *ALLUSR or *IBM is used, the library name must be *LIBL or QSYS and the object type parameter must be *LIB. When QSYS is specified, the list of libraries returned varies with the device name specified in the auxiliary storage pool (ASP) control parameter.

1. When *ALLUSR is specified with a library name of *LIBL and an object type parameter of *LIB, a list of all user libraries in the thread's library name space is returned. When *LIBL is specified, the auxiliary storage pool (ASP) device name must be an asterisk (*) if the auxiliary storage pool (ASP) control parameter is specified. Refer to *ALLUSR in the description of the second 10 characters of this parameter for a definition of user libraries.

2. When *ALLUSR is specified with a library name of QSYS and an object type parameter of

*LIB, a list of all user libraries in the auxiliary storage pools defined by the auxiliary storage pool (ASP) control parameter is returned. Refer to *ALLUSR in the description of the second 10 characters of this parameter for a definition of user libraries.

3. When *IBM is specified with a library name of *LIBL and an object type of *LIB, a list of libraries in the thread's library name space that are saved or restored on the Save Library (SAVLIB) or Restore Library (RSTLIB) CL command with LIB(*IBM) is returned. When *LIBL is specified, the auxiliary storage pool (ASP) device name must be an asterisk (*) if the auxiliary storage pool (ASP) control parameter is specified.

4. When *IBM is specified with a library name of QSYS and an object type of *LIB, a list of libraries in the auxiliary storage pools specified by the auxiliary storage pool (ASP) control parameter that are saved or restored on the Save Library (SAVLIB) or Restore Library (RSTLIB) CL command with LIB(*IBM) is returned.

Library name errors are reported with escape messages when a single library is specified. When searching a set of libraries (library specified as *ALL, *ALLUSR, *LIBL, or *USRLIBL or auxiliary storage pool (ASP) device name specified as *ALLAVL), library errors are reported with diagnostic messages and processing continues. Library authority error messages are not sent when searching a set of libraries. Escape messages are not sent for object name errors. To determine if errors occurred on the object, check the list information returned and the information status field in the receiver variable.

The second 10 characters identify the name of the library or libraries to search for the specified objects. The following special values are allowed:

**ALL* All libraries in the auxiliary storage pools defined by the auxiliary storage pool (ASP) control parameter are searched.

**ALLUSR* All user libraries in the auxiliary storage pools (ASPs) defined by the auxiliary storage pool (ASP) control parameter are searched. User libraries are all libraries with names that do not begin with the letter Q except for the following:

#CGULIB	#RPGLIB
#COBLIB	#SDALIB
#DFULIB	#SEULIB
#DSULIB	

Although the following libraries with names that begin with the letter Q are provided by IBM, they typically contain user data that changes frequently. Therefore, these libraries are also considered user libraries:

QDSNX	QSYS2xxxxx	QUSROND
QGPL	QS36F	QUSRPOSGS
QGPL38	QUSER38	QUSRPOSSA
QMPGDATA	QUSRADSM	QUSRPYMSVR
QMQMDATA	QUSRBRM	QUSRRDARS
QMQMPROC	QUSRDIRCL	QUSRSYS
QPFRDATA	QUSRDIRDB	QUSRVI
QRCL	QUSRIJS	QUSRVxRxMx
QRCLxxxxx	QUSRINFSKR	
QSYS2	QUSRNOTES	

Notes:

1. 'xxxxx' is the number of a primary auxiliary storage pool (ASP).
2. A different library name, of the form QUSRVxRxMx, can be created by

the user for each release that IBM supports. VxRxMx is the version, release, and modification level of the library.

**CURLIB* The thread's current library is searched. When this value is used, the auxiliary storage pool (ASP) device name in the auxiliary storage pool (ASP) control parameter must be an asterisk (*), if specified.

**LIBL* All libraries in the thread's library list are searched. When this value is used, the auxiliary storage pool (ASP) device name in the auxiliary storage pool (ASP) control parameter must be an asterisk (*), if specified.

**USRLIBL* All libraries in the user portion of the thread's library list are searched. When this value is used, the auxiliary storage pool (ASP) device name in the auxiliary storage pool (ASP) control parameter must be an asterisk (*), if specified.

When optional parameter group 1 is specified and the job is not the current job, the library name must be a special value of *CURLIB, *LIBL, or *USRLIBL.

Object type

INPUT; CHAR(10)

The types of objects to search for. You may either enter a specific object type, or the special value *ALL. For a complete list of the available object types, see the Control Language (CL) information.

Authority control

INPUT; CHAR(*)

The authority to check for on objects and libraries. This parameter can be used with the selection control parameter to select the objects to which a user is authorized. To accomplish this, specify a select or omit status value in the selection control parameter. The object name information is always returned for objects meeting the search criteria. (This assumes the job has the required authority to the library.) The information status field is set to an A when the job does not have the object authority that is specified.

The following example shows what you would specify to obtain a subset of all objects that you have object management authority to.

The authority control parameter would contain:

Length of authority control format: 48

Call level: 1

Displacement to object authorities: 28

Number of object authorities: 1

Displacement to library authorities: 38

Number of library authorities: 1

Object authorities: '*OBJMGT '

Library authorities: '*USE '

The selection control parameter would contain:

Length of selection control format: 21

Select or omit status value: 1

Displacement to statuses: 20

Number of statuses: 1

Statuses: 'A'

Because the program that calls the QGYOLOBJ API adopts authority, the authority check should be done at the call level previous to the current level (thus call level 1). With call level 1, the list would not include any objects for which you have adopted authority by the current program.

The select or omit status value of 1 indicates that the returned list will omit the objects you do not have object management authority to. This authority is specified in the object authorities field.

The format of this parameter is described in Authority Control Format.

Selection control

INPUT; CHAR(*)

The criteria used to select or filter objects from the list based on specified information status values.

This parameter is useful to reduce the total number of objects returned in the list. The list of objects can be generated with only the specific status that you are interested in. For example, this might be all damaged objects or all objects that the caller of the API is not authorized to. The list of objects also can be generated with all objects except objects of a specific status.

The following example shows what you would specify to select all damaged objects:

Length of selection control format: 22

Select or omit status value: 0

Displacement to statuses: 20

Number of statuses: 2

Statuses: DP

The format of this parameter is described in Selection Control Format.

Number of keyed fields to return

INPUT; BINARY(4)

The number of keyed fields to return in addition to the fields returned with the specified format.

Key of fields to be returned

INPUT; ARRAY(*) of BINARY(4)

The list of the fields to be returned in addition to the fields returned with the specified format. For a list of the valid keys, see Valid Keys.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter . If this parameter is omitted, diagnostic and escape messages are issued to the application.

Optional Parameter Group 1

Job identification information

INPUT; CHAR(*)

The information that is used to identify the thread within a job for which objects can be searched using the thread's library list. The format of this parameter is described in Job Identification Information Formats.

When this parameter is specified and the job is not the current job, a library name of *CURLIB, *LIBL, or *USRLIBL must be specified for the library name on the object and library name parameter.

A limited amount of information is returned for objects in another job's QTEMP library. Information is returned for keys 201, 202, 203, 205, 701, and 702 only. Any other keys requested will have blanks, hexadecimal 0 (date fields), or 0 returned.

Format of job identification information

INPUT; CHAR(8)

The format of the job identification information. See Job Identification Information Formats for details. The possible format names are:

JIDF0000 See JIDF0000 Format for details on this format. This format is provided to simplify the coding of this parameter when this parameter must be specified because optional parameter groups after this one are to be specified and the functions provided by the other formats of this parameter are not needed. Using this format gives the same result as if optional parameter group 1 were not specified. The information retrieved is for the thread in which this program is running.

JIDF0100 See JIDF0100 Format for details on this format.

JIDF0200 See JIDF0200 Format for details on this format. If the thread handle is available, format JIDF0200 provides faster access than format JIDF0100 to a thread other than the current thread.

Optional Parameter Group 2

Auxiliary storage pool (ASP) control

Input; CHAR(*)

The information used to define the auxiliary storage pool (ASP) to search. The format of this parameter is described in Auxiliary Storage Pool (ASP) Control Format.

Sort Information Format

The following shows the format of the sort information parameter. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Number of keys to sort on
Offsets vary. These fields repeat for each sort key field.		BINARY(4)	Sort key field starting position
		BINARY(4)	Sort key field length
		BINARY(2)	Sort key field data type
		CHAR(1)	Sort order
		CHAR(1)	Reserved.

Note: If the last three fields (sort key field data type, sort order and the reserved field) are not used, they must be set to hexadecimal zeros. This causes all the data to be treated as character data, and it is sorted in ascending order.

Field Descriptions

Number of keys to sort on. The number of fields within the record structure to sort on. If zero is specified, the list is not sorted.

Reserved. An unused. This field must contain hexadecimal zeros.

Sort key field data type. The data type of the field to sort. Refer to the key data type field in the for information about the list of data types available.

Sort key field length. The length of the field to sort on.

Sort key field starting position. The starting position of the field within the record of information to sort on.

Sort order. Whether the list should be sorted in ascending or descending order according to the key.

1 Sort in ascending order.

2 Sort in descending order.

Authority Control Format

The following shows the format of the authority control parameter. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of authority control format

4	4	BINARY(4)	Call level
8	8	BINARY(4)	Displacement to object authorities
12	C	BINARY(4)	Number of object authorities
16	10	BINARY(4)	Displacement to library authorities
20	14	BINARY(4)	Number of library authorities
24	18	BINARY(4)	Reserved
		ARRAY(*) of CHAR(10)	Object authorities
		ARRAY(*) of CHAR(10)	Library authorities

Field Descriptions

Call level. The number of call levels to go back in the call stack to do the authority check.

For example, if the program that calls this API adopts authority, you probably would not want the authority check to use the adopted authority. Therefore, the authority check should be done at the call level previous to the current level. This field should then contain a 1. You can check the authority at various call levels by specifying a number equivalent to the call level. For example, to check the authority at the current call level, specify a 0. To check the authority at the previous call level, specify a 1.

This field must be greater than or equal to 0 and less than the number of programs in the call stack.

Displacement to library authorities. The byte offset from the beginning of the authority control format to the list of library authorities. The offset value must be 0, 28, or a number greater than 28. When 0 is specified, the number of library authorities should also be 0. A value of 28 is past the reserved portion of the format.

Displacement to object authorities. The byte offset from the beginning of the authority control format to the list of object authorities. The offset value must be 0, 28, or a number greater than 28. When 0 is specified, the number of object authorities should also be 0. A value of 28 is past the reserved portion of the format.

Length of authority control format. The total length of the authority control format. The minimum size is 28 bytes. When the size is 28, it is assumed that the number of library authorities and the number of object authorities are both 0. To allow for one object and one library authority, the size should be 48 bytes. An error is returned if the length specified is less than the minimum.

Library authorities. The authority to check for libraries. The array can contain up to ten 10-character fields. If the number of library authorities is 0, *EXECUTE authority is checked for on the libraries.

The authority values can be specified in any combination. If *ALL, *CHANGE, or *USE is specified with any of the other authority values, the authority checked is the cumulative authority value.

The maximum number of authorities that can be specified is 10. This equals all of the specific object and data authorities that can be listed separately.

The following identifies the type of authority the user has to the library:

**ALL* All authority

**CHANGE*	Change authority
**USE*	Use authority
**OBJOPR*	Object operational authority
**OBJMGT*	Object management authority
**OBJEXIST*	Object existence authority
**OBJALTER*	Alter authority
**OBJREF*	Reference authority
**READ*	Read authority
**ADD*	Add authority
**UPD*	Update authority
**DLT*	Delete authority
**EXECUTE*	Execute authority

Number of library authorities. The number of authorities specified in the library authorities array. You can specify 0 through 10 authorities. When 0 is specified, *EXECUTE authority is checked for on the objects.

Number of object authorities. The number of authorities specified in the object authorities array. You can specify 0 through 11 authorities. When 0 is specified, *ANY authority is checked for on the objects.

Object authorities. The authority to check for objects. The array can contain up to eleven 10-character fields. If the number of object authorities is 0, *ANY authority is checked for on the objects.

The authority values can be specified in any combination with the exception of the special value *ANY. This must be specified as the only value. If *ALL, *CHANGE, *USE, or *AUTLMGT is specified with any of the other authority values, the authority checked is the cumulative authority value.

The maximum number of authorities that can be specified is 11, which equals all the specific object and data authorities and *AUTLMGT authority.

The following identifies the type of authority the user has to the object:

**ALL*	All authority
**CHANGE*	Change authority
**USE*	Use authority
**AUTLMGT*	Authorization list management authority. (This value is valid only if the object type is *AUTL. It will be ignored for other object types.)
**OBJOPR*	Object operational authority
**OBJMGT*	Object management authority
**OBJEXIST*	Object existence authority
**OBJALTER*	Alter authority

**OBJREF*	Reference authority
**READ*	Read authority
**ADD*	Add authority
**UPD*	Update authority
**DLT*	Delete authority
**EXECUTE*	Execute authority
**ANY*	Any authority other than *EXCLUDE. (If this value is specified, no other values can be specified.)

Reserved. An unused field. This field must contain hexadecimal zeros.

Selection Control Format

The following shows the format of the selection control parameter. For detailed descriptions of the fields in the table, see Field Descriptions .

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of selection control format
4	4	BINARY(4)	Select or omit status value
8	8	BINARY(4)	Displacement to statuses
12	C	BINARY(4)	Number of statuses
16	10	BINARY(4)	Reserved
		ARRAY(*) of CHAR(1)	Statuses

Field Descriptions

Displacement to statuses. The byte offset from the beginning of the selection control format to the list of statuses requested. The offset value must be at least 20, which is past the reserved portion of the format.

Length of selection control format. The total length of the selection control format. The minimum size is 21 bytes, which allows for one status value. An error is returned if the length specified is less than the minimum.

Number of statuses. The number of statuses specified in the statuses array. You can specify 1 through 5 statuses.

Reserved. An unused field. This field must contain hexadecimal zeros.

Select or omit status value. An indicator that determines whether objects are selected or omitted from the list based on the statuses specified.

This field is useful in generating a list of objects with a certain information status, such as damaged or

partially damaged objects. It can also be used to generate a list of all objects except objects with a certain information status, such as unauthorized objects.

0 Select on status value

1 Omit on status value

Statuses. The status of objects to select or omit from the list of objects generated. Valid values are all of the possible values listed under the information status field in the Format of Receiver Variable. The special value * can be used to select all objects with any information status field.

Job Identification Information Formats

JIDF0000 Format

The following shows the details of format JIDF0000. For detailed descriptions of the field in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Default job name

Field Descriptions

Default job name. This field must contain an asterisk '* '. The information retrieved is for the thread in which this program is running.

JIDF0100 Format

The following shows the details of format JIDF0100 of the information needed to identify the thread's library list used for an object search. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Job name
10	A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(2)	Reserved
44	2C	BINARY(4)	Thread indicator
48	30	CHAR(8)	Thread identifier

Field Descriptions

Internal job identifier. The internal identifier for the job. The List Job (QUSLJOB) API returns this identifier. If you do not specify *INT for the job name parameter, this parameter must contain blanks. With this parameter, the system can locate the job more quickly than with a job name.

Job name. A specific job name or one of the following special values:

*	The job in which this program is running. The job number and user name must contain blanks.
*INT	The internal job identifier locates the job. The job number and user name must contain blanks.

Job number. A specific job number, or blanks when the job name specified is a special value.

Reserved. An unused field. This field must contain hexadecimal zeros.

Thread identifier. The unique value that is used to identify the thread within the job. »If the thread indicator is not 0, this field must contain hexadecimal zeros.«

Thread indicator. The value that is used to specify the thread within the job for which information is to be retrieved. The following values are supported:

» *0*	Specifies that information should be retrieved for the thread specified in the thread identifier field.
1	Specifies that information should be retrieved for the thread that this program is currently running in. The combination of the internal job identifier, job name, job number, and user name fields must also identify the job containing the current thread.«
2	Information should be retrieved for the initial thread of the identified job.

User name. A specific user profile name, or blanks when the job name specified is a special value.

» JIDF0200 Format

The following shows the details of format JIDF0200 for the information needed to identify the thread's library list used for an object search. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Job name
10	A	CHAR(10)	User name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(2)	Reserved
44	2C	BINARY(4), UNSIGNED	Thread handle
48	30	CHAR(8)	Thread identifier

Field Descriptions

Internal job identifier. The internal identifier for the job. The List Job (QUSLJOB) API returns this identifier. If you do not specify *INT for the job name parameter, this parameter must contain blanks. With this parameter, the system can locate the job more quickly than with a job name.

Job name. A specific job name or one of the following special values:

*	The job in which this program is running. The job number and user name must contain blanks.
*INT	The internal job identifier locates the job. The job number and user name must contain blanks.

Job number. A specific job number, or blanks when the job name specified is a special value.

Reserved. An unused field. This field must contain hexadecimal zeros.

Thread handle. A value which is used to address a particular thread within a job. A valid thread handle must be specified. The thread handle is returned on several other interfaces.

Thread identifier. A value which is used to uniquely identify a thread within a job. A valid thread identifier must be specified.

User name. A specific user profile name, or blanks when the job name specified is a special value.

Auxiliary Storage Pool (ASP) Control Format

The following shows the format of the auxiliary storage pool (ASP) control parameter. This parameter is used to define the auxiliary storage pools (ASPs) to search. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of auxiliary storage pool (ASP) control format
4	4	CHAR(10)	Auxiliary storage pool (ASP) device name
14	E	CHAR(10)	Auxiliary storage pool (ASP) search type

Field Descriptions

Auxiliary storage pool (ASP) device name. The name of an auxiliary storage pool (ASP) device in which storage is allocated for the library containing the object. The ASP device must have a status of 'Available'. This parameter must be an asterisk (*) if optional parameter group 2 is specified when *CURLIB, *LIBL, or *USRLIBL is specified as the library name in the object and library name parameter. If optional parameter group 2 is omitted in cases where it is valid for the ASP device name to have a value other than an asterisk (*), the thread's library name space will be used. One of the following special values may be specified:

*	The ASPs in the thread's library name space.

**SYSBAS*	The system ASP (ASP 1) and defined basic user ASPs (ASPs 2-32).
**CURASPGRP*	The ASPs in the current thread's ASP group.
**ALLAVL*	All available ASPs. This includes the system ASP (ASP 1), all defined basic user ASPs (ASPs 2-32), and all available primary and secondary ASPs (ASPs 33-255 with a status of 'Available').

Auxiliary storage pool (ASP) search type. The type of the search when a specific auxiliary storage pool (ASP) device name is specified for the ASP device name field. This field must be blanks when a special value is specified for the auxiliary storage pool (ASP) device name field. One of the following values may be specified:

**ASP*	Only the single ASP named in the auxiliary storage pool (ASP) device name field will be searched.
**ASPGRP*	All ASPs in the auxiliary storage pool (ASP) group named in the auxiliary storage pool (ASP) device name field will be searched. The device name must be the name of the primary auxiliary storage pool (ASP) in the group.

Length of auxiliary storage pool (ASP) control format. The total length of the auxiliary storage pool (ASP) control format. The length can be 0 bytes to indicate that no auxiliary storage pool (ASP) control information is provided. Otherwise, the length must be 24 bytes. An error is returned if the length specified is not 24 or 0.

Format of Receiver Variable

To get all of the information from a format, specify the key that will return all of the fields associated with the format. There is no format parameter for this API.

The offsets given in the tables below apply only if the key for a format is specified. If individual field keys are specified, the fields are returned in the order in which the keys are specified.

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

Offset		Type	Field
Dec	**Hex**		
0	0	CHAR(10)	Object name used
10	A	CHAR(10)	Object library name used
20	14	CHAR(10)	Object type used
30	1E	CHAR(1)	Information status
31	1F	CHAR(1)	Reserved
32	20	BINARY(4)	Number of fields returned
Offsets vary. these fields repeat, in the order listed, for each key field selected.		BINARY(4)	Length of field information returned
		BINARY(4)	Key field for field returned
		CHAR(1)	Type of data
		CHAR(3)	Reserved
		BINARY(4)	Length of data returned
		CHAR(*)	Data

	CHAR(*)	Reserved

Field Descriptions

Data. The actual data contained in the keyed field.

Information status. Whether the QGYOLOBJ API returns the requested information for this object. If you do not request any keys to be returned, ignore this field. Possible values are:

blank The requested information is returned. No errors occurred.

A No information is returned. The job that called this API needs the authority specified in the object authorities field to the object.

D The requested information is returned. However, the object is damaged and should be recreated as soon as possible.

L No information is returned because the object is locked.

P The requested information is returned. However, the object is partially damaged. In most instances, to recover from partial object damage, you delete the damaged object and either restore a saved copy or create the object again. For some damaged objects, special recovery procedures are possible. Refer to the Backup and Recovery book for more information on damaged objects.

If two or more conditions occur that include no authorization (A) to the object, the status is set to A. If the object is damaged (D) and locked (L), or if the object is partially damaged (P) and locked, the status is set to L.

If the value of this field is A or L, your application should not use the other fields for the object. Only the object name, library, and type fields contain accurate data.

Key field for field returned. The specific key for the field being returned. See Valid Keys.

Length of data returned. The length of data returned for this keyed field.

Length of field information returned. The length of information returned for this keyed field.

Number of fields returned. The number of keyed fields of information returned.

Object library name used. The name of the library containing the object.

Object name used. The name of the object.

Object type used. The type of the object. For a list of all the available object types, see the Control Language (CL) information.

Reserved. An unused field. This field contains hexadecimal zeros.

Type of data. The type of data that is contained in this keyed field. Possible values follow:

B Binary data

C Character data

S Special structured field used for key combinations.

Valid Keys

The following table contains a list of the valid keys that are used for the key of fields to be returned parameter. See Field Descriptions for the descriptions of the valid key fields.

Key	Type	Description
0200	CHAR(80)	Special key combination. See Key 0200 Contents for the key combination.
0201	CHAR(1)	Information status
0202	CHAR(10)	Extended object attribute
0203	CHAR(50)	Text description
0204	CHAR(10)	User-defined attribute
0205	BINARY(4)	Order in library list
0300	CHAR(144)	Special key combination. See Key 0300 Contents for the key combination.
0301	BINARY(4)	» Object auxiliary storage pool (ASP) number«
0302	CHAR(10)	Object owner
0303	CHAR(2)	Object domain
0304	CHAR(8)	Creation date and time
0305	CHAR(8)	Change date and time
0306	CHAR(10)	Storage
0307	CHAR(1)	Object compression status
0308	CHAR(1)	Allow change by program
0309	CHAR(1)	Changed by program
0310	CHAR(10)	Object auditing value
0311	CHAR(1)	Digitally signed
» 0312	CHAR(1)	Digitally signed by a system-trusted source
0313	CHAR(1)	Digitally signed more than once
0314	BINARY(4)	Library auxiliary storage pool (ASP) number«
0400	CHAR(296)	Special key combination. See Key 0400 Contents for the key combination.
0401	CHAR(10)	Source file name
0402	CHAR(10)	Source file library name
0403	CHAR(10)	Source file member name
0404	CHAR(13)	Source file updated date and time
0405	CHAR(10)	Creator's user profile
0406	CHAR(8)	System where object was created
0407	CHAR(9)	System level

0408	CHAR(16)	Compiler
0409	CHAR(8)	Object level
0410	CHAR(1)	User changed
0411	CHAR(16)	Licensed program
0412	CHAR(10)	Program temporary fix (PTF)
0413	CHAR(10)	Authorized program analysis report (APAR)
0414	CHAR(10)	Primary group
0500	CHAR(504)	Special key combination. See Key 0500 Contents for the key combination.
0501	CHAR(8)	Object saved date and time
0502	CHAR(8)	Object restored date and time
0503	BINARY(4)	Save size
0504	BINARY(4)	Save size multiplier
0505	BINARY(4)	Save sequence number
0506	CHAR(10)	Save command
0507	CHAR(71)	Save volume ID
0508	CHAR(10)	Save device
0509	CHAR(10)	Save file name
0510	CHAR(10)	Save file library name
0511	CHAR(17)	Save label
0512	CHAR(8)	Save active date and time
0513	CHAR(1)	Journal status
0514	CHAR(10)	Journal name
0515	CHAR(10)	Journal library name
0516	CHAR(1)	Journal images
0517	CHAR(1)	Journal entries to be omitted
0518	CHAR(8)	Journal start date and time
0600	CHAR(548)	Special key combination. See Key 0600 Contents for the key combination.
0601	CHAR(8)	Last-used date and time
0602	CHAR(8)	Reset date and time
0603	BINARY(4)	Days-used count
0604	CHAR(1)	Usage information updated
» 0605	CHAR(10)	Object auxiliary storage pool (ASP) device name
0606	CHAR(10)	Library auxiliary storage pool (ASP) device name«
0700	CHAR(560)	Special key combination. See Key 0700 Contents for the key combination.
0701	BINARY(4)	Object size
0702	BINARY(4)	Object size multiplier
0703	CHAR(1)	Object overflowed ASP indicator

Key 0200 Contents

The following information is returned in the field when key 0200 is specified. For detailed descriptions of the subfields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(1)	Information status
1	1	CHAR(10)	Extended object attribute
11	B	CHAR(50)	Text description
61	3D	CHAR(10)	User-defined attribute
71	47	BINARY(4)	Order in library list
75	4B	CHAR(5)	Reserved

Key 0300 Contents

the following information is returned in the field when key 0300 is specified. For detailed descriptions of the subfields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0		Everything from key 0200
80	50	BINARY(4)	» Object auxiliary storage pool (ASP) number «
84	54	CHAR(10)	Object owner
94	5E	CHAR(2)	Object domain
96	60	CHAR(8)	Creation date and time
104	68	CHAR(8)	Change date and time
112	70	CHAR(10)	Storage
122	7A	CHAR(1)	Object compression status
123	7B	CHAR(1)	Allow change by program
124	7C	CHAR(1)	Changed by program
125	7D	CHAR(10)	Object auditing value
135	87	CHAR(1)	Digitally signed
» 136	88	CHAR(1)	Digitally signed by system-trusted source
137	89	CHAR(1)	Digitally signed more than once
138	8A	CHAR(2)	Reserved
140	8C	BINARY(4)	Library auxiliary storage pool (ASP) number «

Key 0400 Contents

The following information is returned in the field when key 0400 is specified. For detailed descriptions of

the subfields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0		Everything from key 0300
144	90	CHAR(10)	Source file name
154	9A	CHAR(10)	Source file library name
164	A4	CHAR(10)	Source file member name
174	AE	CHAR(13)	Source file updated date and time
187	BB	CHAR(10)	Creator's user profile
197	C5	CHAR(8)	System where object was created
205	CD	CHAR(9)	System level
214	D6	CHAR(16)	Compiler
230	E6	CHAR(8)	Object level
238	EE	CHAR(1)	User changed
239	EF	CHAR(16)	Licensed program
255	FF	CHAR(10)	Program temporary fix (PTF)
265	109	CHAR(10)	Authorized program analysis report (APAR)
275	113	CHAR(10)	Primary group
285	11D	CHAR(11)	Reserved

Key 0500 Contents

The following information is returned in the field when key 0500 is specified. For detailed descriptions of the subfields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0		Everything from key 0400
296	128	CHAR(8)	Object saved date and time
304	130	CHAR(8)	Object restored date and time
312	138	BINARY(4)	Save size
316	13C	BINARY(4)	Save size multiplier
320	140	BINARY(4)	Save sequence number
324	144	CHAR(10)	Save command
334	14E	CHAR(71)	Save volume ID
405	195	CHAR(10)	Save device
415	19F	CHAR(10)	Save file name
425	1A9	CHAR(10)	Save file library name
435	1B3	CHAR(17)	Save label
452	1C4	CHAR(8)	Save active date and time
460	1CC	CHAR(1)	Journal status

461	1CD	CHAR(10)	Journal name
471	1D7	CHAR(10)	Journal library name
481	1E1	CHAR(1)	Journal images
482	1E2	CHAR(1)	Journal entries to be omitted
483	1E3	CHAR(8)	Journal start date and time
491	1EB	CHAR(13)	Reserved«

Key 0600 Contents

The following information is returned in the field when key 0600 is specified. For detailed descriptions of the subfields in the table, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0		Everything from key 0500
» 504	1F8	CHAR(8)	Last-used date and time
512	200	CHAR(8)	Reset date and time
520	208	BINARY(4)	Days-used count
524	20C	CHAR(1)	Usage information updated
525	20D	CHAR(10)	Object auxiliary storage pool (ASP) device name
535	217	CHAR(10)	Library auxiliary storage pool (ASP) device name
545	221	CHAR(3)	Reserved«

Key 0700 Contents

the following information is returned in the field when key 0700 is specified. For detailed descriptions of the subfields in the table, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0		Everything from key 0600
» 548	224	BINARY(4)	Object size
552	228	BINARY(4)	Object size multiplier
556	22C	CHAR(1)	Object overflowed ASP indicator
557	22D	CHAR(3)	Reserved«

Field Descriptions

Allow change by program. Whether the object can be changed by the Change Object Description (QLICOBJD) API.

0 The object cannot be changed with the QLICOBJD API.

1 The object can be changed with the QLICOBJD API.

Authorized program analysis report (APAR). The identifier of the authorized program analysis report (APAR) that caused this object to be replaced. the field is blank if the object did not change because of an APAR.

Changed by program. Whether the object has been changed by the Change Object Description (QLICOBJD) API.

0 The object has not been changed with the QLICOBJD API.

1 The object has been changed with the QLICOBJD API.

Change date and time. The time at which the object was last changed, in system time-stamp format.

Compiler. The licensed program identifier, version number, release level, and modification level of the compiler. The field has a ppppppVvvRrrMmm format, where:

ppppppp	The licensed program identifier.
Vvv	The character V is followed by a 2-character version number.
Rrr	The character R is followed by a 2-character release level.
Mmm	The character M is followed by a 2-character modification level.

The field is blank if you do not compile the program.

Creation date and time. The time at which the object was created, in system time-stamp format. See the Convert Date and Time Format (QWCCVTDT) API for more information about using this time-stamp format.

Creator's user profile. The name of the user that created the object.

Days-used count. The number of days the object was used. If the object does not have a last-used date, the count is 0.

Digitally signed. A 1-character variable that indicates whether the object has an OS/400 digital signature.

0 The object does not have an OS/400 digital signature.

1 The object has an OS/400 digital signature.

Digitally signed by system-trusted source. A 1-character variable indicates whether the object is signed by a source that is trusted by the system.

0 None of the object signatures came from a source that is trusted by the system.

1 The object is signed by a source that is trusted by the system. If the object has multiple signatures, at least one of the signatures came from a source that is trusted by the system.

Digitally signed more than once. A 1-character variable that indicates whether the object has more than one OS/400 digital signature.

0 The object has only one OS/400 digital signature or does not have an OS/400 digital signature. Refer to the digitally signed variable to determine whether the object has an OS/400 digital signature.

1 The object has more than one OS/400 digital signature. Refer to the digitally signed by system-trusted source variable to determine whether the object has an OS/400 digital signature from a source trusted by the system.

Extended object attribute. The extended attribute of the object, such as a program or file type. Extended attributes further describe the object. For example, an object type of *PGM may have a value of RPG (RPG program) or CLP (CL program), and an object type of *FILE may have a value of PF (physical file), LF (logical file), DSPF (display file), SAVF (save file), and so on.

Information status. Whether the QGYOLOBJ API returns the requested information for this object. If you do not request any keys to be returned, ignore this field. Possible values are:

blank	The requested information is returned. No errors occurred.
A	No information is returned. The job that called this API needs the authority specified in the object authorities field to the object.
D	The requested information is returned. However, the object is damaged and should be recreated as soon as possible.
L	No information is returned because the object is locked.
P	The requested information is returned. However, the object is partially damaged. In most instances, to recover from partial object damage, you delete the damaged object and either restore a saved copy or create the object again. For some damaged objects, special recovery procedures are possible. Refer to the Backup and Recovery book for more information on damaged objects.

If two or more conditions occur that include no authorization (A) to the object, the status is set to A. If the object is damaged (D) and locked (L), or if the object is partially damaged (P) and locked, the status is set to L.

If the value of this field is A or L, your application should not use the other fields for the object. Only the object name, library, and type fields contain accurate data.

Journal entries to be omitted. The journal entries to be omitted. the field is 1 if *open* and *close* operations do not generate *open* and *close* journal entries. The field is 0 if no entries are omitted. This field is blank if the object has never been journaled.

Journal images. The type of images that are written to the journal receiver for updates to the object. the field is 0 if only *after* images are generated for changes to the object. The field is 1 if both *before* and *after* images are generated for changes to the object. This field is blank if the object has never been journaled.

Journal library name. The name of the library containing the journal. This field is blank if the object has never been journaled.

Journal name. The name of the current or last journal. This field is blank if the object has never been journaled.

Journal start date and time. The time at which journaling for the object was last started, in system time-stamp format. This field contains hexadecimal zeros if the object has never been journaled.

Journal status. The 1-character variable that returns the current journaling status of an object. The value is 1 if the object currently is being journaled; the value is 0 if the object currently is not being journaled.

Last-used date and time. The date and time at which the object was last used, in system time-stamp format. If the object has no last-used date, the field contains hexadecimal zeros.

»**Library auxiliary storage pool (ASP) device name.** The name of the auxiliary storage pool (ASP) device where storage is allocated for the library containing the object. The following special value may be returned:

**SYSBAS* System ASP (ASP 1) or defined basic user ASPs (ASPs 2-32)

Library auxiliary storage pool (ASP) number. The number of the auxiliary storage pool (ASP) where storage is allocated for the library containing the object. Valid values are:

1 System ASP

2-32 Basic user ASP

33-255 Primary or secondary ASP«

Licensed program. The name, release level, and modification level of the licensed program if the retrieved object is part of a licensed program. The 7-character name starts in character position 1, the version number starts in position 8, the release level starts in position 11, and the modification level starts in position 14. The field is blank if the retrieved object is not a part of a licensed program.

Object auditing value. The type of auditing for an object. The valid values are:

**NONE* No auditing occurs for this object when it is read or changed regardless of the user who is accessing the object.

**USRPRF* Audit this object only if the user accessing the object is being audited. The user profile for the job is tested to determine if auditing should be done for this object. The user profile can specify if only change access is audited or if both read and change accesses are audited for this object.

**CHANGE* Audit all change access to this object by all users on the system.

**ALL* Audit all access to this object by all users on the system. All access is defined as a read or change operation.

»**Object auxiliary storage pool (ASP) device name.** The name of the auxiliary storage pool (ASP) device where storage is allocated for the object. The following special value may be returned:

**SYSBAS* System ASP (ASP 1) or basic user ASPs (ASPs 2-32)

Object auxiliary storage pool (ASP) number. The auxiliary storage pool (ASP) identifier where storage is allocated for the object. Valid values are:

1 System ASP

2-32 Basic user ASP

33-255 Primary or secondary ASP«

Object compression status. Whether the object is compressed or decompressed. The status is returned with one of these values:

Y Compressed.

N Permanently decompressed and compressible.

X Permanently decompressed and *not* compressible.

T Temporarily decompressed.

F Saved with storage freed; compression status cannot be determined.

Temporarily decompressed objects exist in both decompressed and compressed form. *Permanently* decompressed objects exist in decompressed form only. the system handles some decompression automatically, depending on the type of object, the operation performed on it, and its frequency of use. For an overview of object compression and decompression, see theCL Programming book. For details about how to explicitly compress and decompress objects, see the entries for these commands in the Control Language (CL) information: Compress Object (CPROBJ), Decompress Object (DCPOBJ), and Reclaim Temporary Storage (RCLTMPSTG).

Object domain. The domain that contains the object. Possible values follow:

**S* The object is in the system domain.

**U* The object is in the user domain.

Object level. The object control level for the created object.

Object overflowed ASP indicator. Whether the object overflowed the auxiliary storage pool (ASP).

0 The object has not overflowed the ASP.

1 The object overflowed the ASP in which it resides.

For objects in the system » auxiliary storage pool (ASP 1) or in a primary or secondary ASP (ASPs 33-255), a 0 is always returned because it is not possible for an object that resides in the system ASP or in a primary or secondary ASP« to overflow its ASP.

Object owner. The name of the object owner's user profile.

Object restored date and time. The time at which the object was restored, in system time-stamp format. If the object has never been restored, the field contains hexadecimal zeros.

Object saved date and time. The time at which the object was saved, in system time-stamp format. If the object has never been saved, the field contains hexadecimal zeros.

Object size. The size of the object in units of the size multiplier. The object size is equal to or smaller than the object size multiplied by the object size multiplier.

Object size multiplier. The value to multiply the object size by to get the true size. The value is 1 if the object is smaller than 1 000 000 000 bytes, and 1024 if it is larger.

Order in library list. The order in which the library appears in the entire library list. If the library is in the library list more than once, the order of the first occurrence of the library is returned. If the library is not in the library list, then 0 is returned for the order number.

Primary group. The name of the user who is the primary group for the object. If no primary group exists for the object, this field contains a value of *NONE.

Program temporary fix (PTF). The number of the program temporary fix (PTF) number that caused this object to be replaced. This field is blank if the object was not changed because of a PTF.

Reserved. An unused field. This field contains hexadecimal zeros.

Reset date and time. The date the days-used count was last reset to zero, in system time-stamp format. If the days-used count has never been reset, the field contains hexadecimal zeros.

Save active date and time. The date and time the object was last saved when the SAVACT(*LIB, *SYSDFN, or *YES) save operation was specified, in system time-stamp format. This parameter is found on the Save Library (SAVLIB), Save Object (SAVOBJ), Save Changed Object (SAVCHGOBJ), and Save Document Library Object (SAVDLO) CL commands. If the object has never been saved or if SAVACT(*NO) was specified on the last save operation for the object, the field contains hexadecimal zeros.

Save command. The command used to save the object. The field is blank if the object was not saved.

Save device. The type of device to which the object was last saved. The field is *SAVF if the last save operation was to a save file. The field is *DKT if the last save operation was to diskette. The field is *TAP if the last save operation was to tape. The field is *OPT if the last save operation was to optical. The field is blank if the object was not saved.

Save file library name. The name of the library that contains the save file if the object was saved to a save file. The field is blank if the object was not saved to a save file.

Save file name. The name of the save file if the object was saved to a save file. The field is blank if the object was not saved to a save file.

Save label. The file label used when the object was saved. The variable is blank if the object was not saved to tape, diskette, or optical. The value of the variable corresponds to the value specified for the LABEL parameter on the command used to save the object.

Save sequence number. The tape sequence number assigned when the object was saved on tape. The field contains zeros if the object was not saved.

Save size. The size of the object in bytes of storage at the time of the last save operation. The save size is equal to or smaller than the save size multiplied by the save size multiplier. The field contains zeros if the object was not saved.

Save size multiplier. The value to multiply the save size by to get the true size. The value is 1 if the save size is smaller than 1 000 000 000 bytes, and 1024 if it is larger.

Save volume ID. The tape, diskette, or optical volumes that are used for saving the object. The variable returns a maximum of ten 6-character volumes. The volume IDs begin in character positions 1, 8, 15, 22, 29, 36, 43, 50, 57, and 64. Each volume ID entry is separated by a single character. If the object was saved in parallel format, the separator character contains a 2 before the first volume in the second media file, a 3 before the third media file, and so on, up to a 0 before the tenth media file. Otherwise, the separator characters are blank. If more than 10 volumes are used and the object was saved in serial format, 1 is returned in the 71st character of the variable. If the object was saved in parallel format, a 2 is returned in the 71st character of the variable. Otherwise, the 71st character is blank. The field is blank if the object was last saved to a save file or if it was never saved.

Source file library name. The name of the library that contains the source file that is used to create the object. The field is blank if no source file created the object.

Source file member name. The name of the member in the source file. The field is blank if no source file created the object.

Source file name. The name of the source file that is used to create the object. The field is blank if no source file created the object.

Source file updated date and time. The date and time the member in the source file was last updated. This field is in the CYYMMDDHHMMSS format:

C	Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
YY	Year
MM	Month
DD	Day
HH	Hour
MM	Minute
SS	Second

The field is blank if no source file created the object.

Storage. The storage status of the object data. *FREE indicates the object data is freed and the object is suspended. *KEEP indicates the object data is not freed and the object is not suspended.

System level. The level of the operating system when the object was created. The field has a VvvRrrMmm format, where:

Vvv	The character V is followed by a 2-character version number.
Rrr	The character R is followed by a 2-character release level.
Mmm	The character M is followed by a 2-character modification level.

System where object was created. The name of the system on which the object was created.

Text description. The text description of the object. The field is blank if no text description is specified.

Usage information updated. Whether the object usage information is updated for this object type. The indicator is returned as Y (Yes) or N (No).

User changed. Whether the user program was changed.

0	The object was not changed by the user.
1	The user changed the object.

User-defined attribute. A characteristic of an object type. This field is set by the user while using the Change Object Description (QLICOBJD) API.

Error Messages

Message ID	Error Message Text
» CPFB8ED E	Device description &1 not correct for operation.«
CPF136A E	Job &3/&2/&1 not active.
» CPF18BF E	Thread &1 not found.«
CPF1867 E	Value &1 in list not valid.
CPF21AA E	Number of statuses must be between 1 and 5.
CPF21AB E	Status value &1 not valid.
CPF21AC E	Length or offset value &1 not valid.
CPF21A7 E	Authority value &1 not valid.
CPF21A8 E	Must specify *ANY as only authority value.
CPF21A9 E	Select or omit value &1 not valid.
» CPF218C E	&1 not a primary or secondary ASP.«
» CPF218D E	&1 not a primary ASP when *ASPGRP specified.«
CPF22F7 E	Number of authorities must be between 1 and &1.
CPF22F9 E	Call level &1 not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C3A E	Value for parameter &2 for API &1 not valid.
CPF3C3B E	Value for parameter &2 for API &1 not valid.
CPF3C31 E	Object type &1 is not valid.
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C51 E	Internal job identifier not valid.
CPF3C52 E	Internal job identifier no longer valid.
CPF3C53 E	Job &3/&2/&1 not found.
CPF3C55 E	Job &3/&2/&1 does not exist.
CPF3C57 E	Not authorized to retrieve job information.
CPF3C58 E	Job name specified is not valid.

CPF3C59 E	Internal identifier is not blanks and job name is not *INT.
CPF3C90 E	Literal value cannot be changed.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF9804 E	Object &2 in library &3 damaged.
CPF9810 E	Library &1 not found.
» CPF9814 E	Device &1 not found.«
CPF9820 E	Not authorized to use library &1.
» CPF9825 E	Not authorized to device &1.«
CPF9830 E	Cannot assign library &1.
» CPF9833 E	*CURASPGRP or *ASPGRPPRI specified and thread has no ASP group.«
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.
GUI0024 E	&1 is not valid for number of keys to sort on.
GUI0025 E	&1 is not valid for sort key field starting position.
GUI0026 E	&1 is not valid for sort key field length.
GUI0027 E	&1 is not valid for number of records to return.
GUI0083 E	&1 is not valid for number of fields to return.

API introduced: V3R6

Top | Object APIs | APIs by category

Rename Object (QLIRNMO) API

Required Parameter Group:

1	From qualified object name	Input	Char(20)
2	Object type	Input	Char(10)
3	To qualified object name	Input	Char(20)
4	Replace object	Input	Char(1)
5	Error code	I/O	Char(*)

»Optional Parameter Group 1:

6	From library auxiliary storage pool (ASP) device name	Input	Char(10)
7	To library auxiliary storage pool (ASP) device name	Input	Char(10)«

Default Public Authority: *USE

Threadsafe: Conditional; see Usage Notes.

The Rename Object (QLIRNMO) API renames an existing object to a new object name or moves the object to a different library or both, and optionally replaces the object.

When the replace object parameter requests to replace an existing object and the to object already exists, the following occur:

- The *PUBLIC and private authorities from the to object replace the authorities on the renamed object.
- The owner of the object is the owner of the from object.
- If the object is not a *PGM object, the to object is deleted.
- »For a *PGM object, the to object is moved to the QRPLOBJ library (or the QRPLxxxxx library if the to object is in a library in primary or secondary auxiliary storage pool 'xxxxx').« The renamed program will have the same user profile (USRPRF) value as the to program. If the to program has the adopted authority USRPRF(*OWNER) attribute, the owner of the from program must be the same as the owner of the to program. An error message is issued if the owners do not match. For more information on adopted authority, see the iSeries Security Reference book. The use adopted authority (USEADPAUT) value from the to program is copied to the from program as long as the user who performs the rename operation can create and update programs with the USEADPAUT(*YES) attribute. The QUSEADPAUT system value determines whether or not users can create and update programs to use adopted authority. If the program being replaced has USEADPAUT(*YES) and the user cannot create and update programs to use adopted authority, the USEADPAUT value of the from program remains the same.

- For a *CRQD object, the from change request description will have the same user profile (USRPRF) value as the to change request description. If the to change request description has a user profile value of USRPRF(*OWNER), the owner of the from object must be the same as the to object. An error message is issued if the owners do not match.

Restrictions

All restrictions that apply to the Move Object (MOVOBJ) and Rename Object (RNMOBJ) commands also apply to the QLIRNMO API.

Authorities and Locks

Auxiliary Storage Pool (ASP) Device Authority

*USE when a specific ASP device name is specified for optional parameter group 1.

Library Authority

*CHANGE

Note: If you are renaming an object that can only exist in library QSYS, the library authority is not checked.

Object Authority

*OBJMGT

Notes:

1. Object types of *FILE, *JRN, *JRNRCV, and *MSGQ need *OBJOPR and *OBJMGT authorities.
2. An object type of *AUTL needs *AUTLMGT authority.

Library Lock

*SHRUPD

Object Lock

*EXCL

If you replace an object, you must be the owner of the from object or have *ALLOBJ special authority. *ALLOBJ authority is needed to replace the authority on the from object.

When the request is to replace an existing object and the to object already exists, the following authority considerations apply:

- For *PGM objects, the user must have *OBJEXIST, *OBJMGT, and *READ authorities to the existing program object.
- For *MENU and *FILE objects, *OBJOPR and *OBJEXIST authorities are needed in addition to the other authorities listed.
- For other object types, *OBJEXIST authority is needed to delete the existing object in addition to the other authorities listed. *OBJEXIST and *USE authorities are needed to delete *LIB and *SBSD objects.

Required Parameter Group

From qualified object name

INPUT; CHAR(20)

The object being renamed and the library in which it is located. The first 10 characters contain the object name, and the second 10 characters contain the library name.

You can use these special values for the library name:

**CURLIB* » The thread's current library«

**LIBL* » The thread's library list«

Object type

INPUT; CHAR(10)

The type of object being renamed. If the from object and the to object belong to the same library, only a rename operation is done. The object type must be supported on the RNMOBJ command. If the from object and the to object belong to different libraries, a move operation is done. The object type must be supported on the MOVOBJ command. If both a rename and a move operation are done, the object type must be supported on both the RNMOBJ and MOVOBJ commands. An asterisk (*) must precede the object type. For a list of the object types that cannot be moved or renamed, see the CL Programming book.

To qualified object name

INPUT; CHAR(20)

The new name of the object and the new library in which it will be located. The object name can be the same name as the original object or a new name. The library name can be the same name as the original library or a new name. If the object name is the same as the original object, the library name must not be the same as the original library »unless the to and from auxiliary storage pool (ASP) device names are not the same.«The first 10 characters contain the object name, and the second 10 characters contain the library name.

Replace object

INPUT; CHAR(1)

Whether to replace an existing object with the same name as the to object and library name parameter »in the auxiliary storage pool named in the to auxiliary storage pool (ASP) device name parameter. «The following values can be specified:

0 Do not replace the existing object. If 0 is specified and the object already exists, an error message is returned to the application.

1 Replace the existing object.

Error code

I/O; CHAR(*)

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

Optional Parameter Group 1

From library auxiliary storage pool (ASP) device name

INPUT; CHAR(10)

The name of the auxiliary storage pool (ASP) device from which storage is allocated for the library containing the object to be renamed or moved. For all operations other than a library rename, the ASP device must have status of 'Available'. For a library rename operation, the ASP device must have a status of 'Active' or 'Available' and the to library auxiliary storage pool (ASP) device and the from auxiliary storage pool (ASP) device must be the same device. If this parameter is omitted in cases where it is valid for this parameter to have a value other than an asterisk (*), the thread's library name space will be used.

This parameter must be an asterisk (*) if specified when *CURLIB or *LIBL is specified as the library name in the from qualified object name parameter. If a library to be renamed is in an auxiliary storage pool (ASP) device that is not currently part of the thread's library name space, specify *current-library-name* in the first 10 characters and QSYS in the second 10 characters of the from qualified object name and the *auxiliary-storage-pool-device-name* in the from library auxiliary storage pool (ASP) device name.

One of the following special values may be specified:

*	The ASPs in the thread's library name space.
**CURASPGRP*	The ASPs in the thread's ASP group.
**SYSBAS*	The system ASP (1) and defined basic user ASPs (2-32).

To library auxiliary storage pool (ASP) device name

INPUT; CHAR(10)

The name of the auxiliary storage pool (ASP) device from which storage is allocated for the library to contain the object after the rename or move. For all operations other than a library rename, the ASP device must have status of 'Available'. For a library rename operation, the ASP device must have a status of 'Active' or 'Available' and the to library auxiliary storage pool (ASP) device and the from auxiliary storage pool (ASP) device must be the same device. If this parameter is omitted, the thread's library name space will be used.

If a library to be renamed is to be in an auxiliary storage pool (ASP) device that is not currently part of the thread's library name space, specify *new-library-name* in the first 10 characters and QSYS in the second 10 characters of the to qualified object name and specify either *SAME or an *auxiliary-storage-pool-device-name* in the to library auxiliary storage pool (ASP) device name that is the same as specified in the from library auxiliary storage pool (ASP) device name.

One of the following special values may be specified:

*	The ASPs in the thread's library name space.
**SAME*	The library to which the object will be renamed or moved is in the same ASP as the library containing the object to be renamed or moved.

**CURASPGRP*	The ASPs in the thread's ASP group.
**SYSBAS*	The system ASP (ASP 1) and defined basic user ASPs (ASPs 2-32).«

Usage Notes

This API is conditionally threadsafe. For multithreaded jobs, see the restrictions in the Rename Object (RNMOBJ) command.

Error Messages

Message ID	Error Message Text
» CPFB8ED E	Device description &1 not correct for operation.«
CPF180B E	Function &1 not allowed.
CPF21A3 E	&1 not valid for replace object option.
CPF21A4 E	Objects cannot be moved into QTEMP.
CPF21A5 E	Cannot replace object &1 in &2 type *&3.
CPF2111 E	Library &1 already exists.
CPF2112 E	Object &1 in &2 type *&3 already exists.
CPF2132 E	Object &1 already exists in library &2.
CPF2136 E	Renaming library &1 failed.
CPF2139 E	Rename of library &1 failed.
CPF2140 E	Rename of library &1 previously failed.
CPF2146 E	Owner of object &1 and object being replaced not the same.
CPF2160 E	Object type *&1 not eligible for requested function.
CPF2164 E	Rename of library &2 not complete.
CPF2166 E	System library cannot be renamed or deleted.
» CPF2173 E	Value for ASPDEV not valid with special value for library.«
CPF2176 E	Library &1 damaged.
» CPF218C E	&1 not a primary or secondary ASP.«
CPF2183 E	Object &1 cannot be moved into library &3.
CPF2189 E	Not authorized to object &1 in &2 type *&3.
CPF219E E	Object type *&1 not valid external object type.

CPF2193 E	Object &1 cannot be moved into library &4.
CPF22BC E	Object &1 type &3 is not program defined.
CPF24B4 E	Severe error while addressing parameter list.
CPF2512 E	Operation not allowed for message queue &1.
CPF2691 E	Rename of &2 type *&5 did not complete.
CPF2692 E	Object &2 type *&5 must be varied off.
CPF2693 E	&2 type *&5 cannot be used for rename.
CPF2694 E	Object &2 type *&5 cannot be renamed.
» CPF3C3A E	Value for parameter &2 for API &1 not valid.«
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF7010 E	Object &1 in &2 type *&3 already exists.
CPF88C4 E	Value &1 for new object is more than 8 characters.
CPF9801 E	Object &2 in library &3 not found.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9809 E	Library &1 cannot be accessed.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9812 E	File &1 in library &2 not found.
CPF9814 E	Device &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9821 E	Not authorized to program &1 in library &2.
» CPF9825 E	Not authorized to device &1.«
CPF9830 E	Cannot assign library &1.
CPF9831 E	Cannot assign device &1.
» CPF9833 E	*CURASPGRP or *ASPGRPPRI specified and thread has no ASP group.«
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V2R3

Retrieve Library Description (QLIRLIBD) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Library name	Input	Char(10)
4	Attributes to retrieve	Input	Char(*)
5	Error code	I/O	Char(*)

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve Library Description (QLIRLIBD) API lets you retrieve attributes for a specific library.

Authorities and Locks

Library Authority

*READ

Library Lock

*SHRRD

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable that is to receive the information requested. If this area is smaller than the actual length of the data returned, the API returns only the data that the area can hold. Refer to Format of Data Returned for details about the format.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable. The minimum length is 8 bytes. If the length is larger than the size of the receiver variable, results may be unpredictable.

Library name

INPUT; CHAR(10)

The name of the library for which information is being retrieved.

Attributes to retrieve

INPUT; CHAR(*)

The information for the library that you want to retrieve.

The information must be in the following format:

Number of elements in request array	BINARY(4) The total number of all of the request keys.
Request keys	ARRAY of BINARY(4) An array of request keys to identify what fields of information about the library are requested. The size of the array is defined in the preceding number of elements in request array value. For a list of the valid key identifiers, see the topic Keys.

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 Data Returned

For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Variable length records returned
12	C	BINARY(4)	Variable length records available
16	10	CHAR(*)	Variable length record for each key specified. For the specific format of the variable length record, see Format for Variable Length Record.

Format for Variable Length Record

For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of returned data
4	4	BINARY(4)	Key identifier

8	8	BINARY(4)	Size of field
12	C	CHAR(*)	Field value
		CHAR(*)	Reserved

Field Descriptions

Bytes available. The length of all data available to return. All available data is returned if enough space is provided.

Bytes returned. The length of the data actually returned. This value includes the length of this field and all the fields following it in the structure.

If insufficient space is provided for the receiver variable, this value would be set to the last byte of the last complete variable length record.

Field value. The value of the field returned.

Key identifier. The key that identifies the returned field. For a list of the valid keys, see Keys.

Length of returned data. The length associated with a particular field.

The length includes the space required to hold the following fields:

- This field
- The key identifier
- The size of (returned) field
- The field value

Reserved. An unused field. This field contains hexadecimal zeros. If multiple keys are requested, a reserved value is added for boundary alignment.

Size of field. The size of the returned field.

Variable length records available. The number of complete variable length records that can be returned. All variable length records are returned if enough space is provided.

Variable length records returned. The number of variable length records actually returned.

Keys

The following table lists the valid key identifiers that can be specified in the attributes to retrieve parameter. See the Field Descriptions for the descriptions of the valid key fields.

Key ID	Type	Field
1	CHAR(1)	Type of library
2	BINARY(4)	Auxiliary storage pool (ASP) number
3	CHAR(10)	Create authority
4	CHAR(10)	Create object auditing

5	CHAR(50)	Text description
6	CHAR(12)	Library size information
7	BINARY(4)	Number of objects in library
8	CHAR(10)	Auxiliary storage pool (ASP) device name

Field Descriptions

Auxiliary storage pool (ASP) device name. The name of the auxiliary storage pool (ASP) device from which the system allocates storage for the library. The following special values may be returned:

**N* The name of the ASP device cannot be determined.

**SYSBAS* System ASP (ASP 1) or basic user ASPs (ASPs 2-32).

Auxiliary storage pool (ASP) number. The number of the auxiliary storage pool (ASP) from which the system allocates storage for the library.

Possible values are:

1 System ASP

2-32 Basic user ASPs

33-255 Primary or secondary ASPs

Create authority. The default public authority used when an object is created into a library. This authority is given to the following users:

- Users who do not have specific authority to the object.
- Users who are not on the authorization list.
- Users whose user group has no specific authority to the object.

The valid values are:

**ALL*	The user can perform all authorized operations on an object created in this library.
**CHANGE*	The user can read the object description and has read, add, update, and delete authority to an object created in this library.
**EXCLUDE*	The user is prevented from accessing an object created in this library.
**SYSVAL*	The default authority for an object created in this library is determined by the value specified by the QCRTAUT system value.
**USE*	The user can read the object and its description but cannot change them for an object created in this library.
Authorization list name	The name of the authorization list that secures an object created in this library. The default public authority is taken from the authorization list, and the public authority for the object is specified as *AUTL.

Create object auditing. The auditing value for objects created in this library.

The valid values are:

**ALL*	All change or read access to the object is logged.
**CHANGE*	All change access to the object by all users is logged.
**NONE*	Use or change access to the object is not logged (no audit entry is sent to the security journal).
**SYSVAL*	The value specified in the system value QCRTOBJAUD is used.
**USRPRF*	The user profile of the user who accesses the object is used to determine if an audit record is sent for this access. The OBJAUD parameter of the Change User Auditing (CHGUSRAUD) command is used to turn auditing on for a specific user.

Library size information. Information about the size of the library. To include the size of an object in the total library size, you need an authority other than *EXCLUDE to the object. See Library Size Information Format for the format of this key.

Number of objects in library. The total number of objects in the specified library.

Text description. The user-defined text that briefly describes the library and its function.

Type of library. The library type.

Possible values are:

0 The library is a production library. Database files in production libraries cannot be opened for updating if a user, while in debug mode, requested that production libraries be protected.

1 The library is a test library. All objects in a test library can be updated during a test. See the Start Debug (STRDBG) command in the online help for more details.

Library Size Information Format

The following table shows the layout of the library size information key. For detailed descriptions of the fields, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0	BINARY(4)	Library size
4	4	BINARY(4)	Library size multiplier
8	8	CHAR(1)	Information status
9	9	CHAR(3)	Reserved

Field Descriptions

Information status. Whether or not all objects in the library were calculated in the library size.

The following values can be returned:

0 Some objects in the library are locked, or the user does not have any authority to the object. The size of these objects was not included in the total library size.

1 The size of all the objects in the library was used in determining the total library size.

Library size. The size of the library object and all of the objects in the library in units of the library size multiplier. If the information status field is 1, the total library size is equal to or smaller than the library size multiplied by the library size multiplier. If the information status field is 0, the total library size could be greater than the library size multiplied by the library size multiplier because the size of some objects has not been included in the total library size.

Library size multiplier. The value to multiply the library size by to get the total library size.

The following values can be returned:

1 The total library size is smaller than 1, 000, 000, 000 bytes.

1024 The total library size is between 1, 000, 000, 000 and 1, 024, 000, 000, 000 bytes.

1 048 576 The total library size is larger than 1, 024, 000, 000, 000 bytes.

Reserved. An unused field. This field contains hexadecimal zeros.

Error Messages

Message ID	Error Message Text
CPFB8ED E	Device description &1 not correct for operation.
CPF2115 E	Object &1 in &2 type *&3 damaged.
CPF2150 E	Object information function failed.
CPF2151 E	Operation failed for &2 in &1 type *&3.
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C82 E	Key &1 not valid for API &2.
CPF3C88 E	Number of variable length records &1 is not valid.
CPF3C89 E	Key &1 specified more than once.

CPF3C90 E	Literal value cannot be changed.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
»CPF980B E	Object &1 in library &2 not available.«
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V3R1

Top | Object API categories | API by category

Retrieve Object Description (QUSROBJD) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Object and library name	Input	Char(20)
5	Object type	Input	Char(10)

»Optional Parameter Group 1:«

6	Error code	I/O	Char(*)

»Optional Parameter Group 2:

7	Auxiliary storage pool (ASP) control	Input	Char(*)«

Default Public Authority: *USE

Threadsafe: Yes

The Retrieve Object Description (QUSROBJD) API lets you retrieve object information about a specific object. This information is similar to the information returned using the Display Object Description (DSPOBJD) command » or Retrieve Object Description (RTVOBJD) command.«

You can use the QUSROBJD API to:

- Determine who owns which objects in the specified libraries
- Provide disk management functions based on the object's size and use
- Provide backup analysis based on when the object was last saved or last updated
- Provide source member and object analysis from source member information to verify that the current source was used to create the specified object
- Work with a list of objects created by the QUSLOBJ API

Authorities and Locks

»*Auxiliary Storage Pool (ASP) Device Authority*

*EXECUTE when a specific auxiliary storage pool (ASP) device name or *ALLAVL is specified for the auxiliary storage pool (ASP) control parameter.«

Library Authority

*EXECUTE

*Object Authority for Non-*FILE Objects*

Any authority other than *EXCLUDE

*Object Authority for *FILE Objects*

*OBJOPR

Library Lock

None

Object Lock

*SHRRD

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable that is to receive the requested information. It can be smaller than the format requested as long as the next parameter, length of receiver variable, specifies the length correctly. When this variable is smaller than the format, the API returns only the data that the variable can hold.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable. The minimum length is 8 bytes. Do not specify a length that is longer than the receiver variable; the results are unpredictable.

Format name

INPUT; CHAR(8)

The content and format of the information returned for each specified member. The possible format names are:

OBJD0100	Basic information (fastest)
OBJD0200	Information similar to that displayed by the programming development manager (PDM)
OBJD0300	Service information
OBJD0400	Full information (slowest)

These are described in the following sections.

Object and library name

INPUT; CHAR(20)

The object for which you want to retrieve information, and the library in which it is located. The first 10 characters contain the object name, and the second 10 characters contain the library name. You can use these special values for the library name:

*CURLIB » The thread's current library is searched. When this value is used, the auxiliary storage pool (ASP) device name in the auxiliary storage pool (ASP) control parameter must be an asterisk (*), if specified.

*LIBL All libraries in the thread's library list are searched. When this value is used, the auxiliary storage pool (ASP) device name in the auxiliary storage pool (ASP) control parameter must be an asterisk (*), if specified.«

Object type

INPUT; CHAR(10)

The type of object for which you want to retrieve the information. You can only specify external object types. Refer to the Control Language (CL) information in the iSeries Information Center for a complete list of available object types.

»Optional Parameter Group 1«

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter. If this parameter is omitted, diagnostic and escape messages are issued to the application.

»Optional Parameter Group 2

Auxiliary storage pool (ASP) control

Input; CHAR(*)

The information used to define the auxiliary storage pool (ASP) to search. See Auxiliary Storage Pool (ASP) Control Format for details. If optional parameter group 2 is omitted in cases where it is valid for the ASP device name to have a value other than an asterisk (*), the thread's library name space will be used. «

»Auxiliary Storage Pool (ASP) Control Format

The following shows the format of the auxiliary storage pool (ASP) control parameter. This parameter is used to define the auxiliary storage pools (ASPs) to search. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of auxiliary storage pool (ASP) control format
4	4	CHAR(10)	Auxiliary storage pool (ASP) device name

14	E	CHAR(10)	Auxiliary storage pool (ASP) search type

Field Descriptions

Auxiliary storage pool (ASP) device name. The name of an auxiliary storage pool (ASP) device in which storage is allocated for the library containing the object. The ASP device must have a status of 'Available'. This field must be an asterisk (*) if optional parameter group 2 is specified when *CURLIB or *LIBL is specified as the library name in the object and library name parameter. If optional parameter group 2 is omitted in cases where it is valid for the ASP device name to have a value other than an asterisk (*), the thread's library name space will be used. One of the following special values may be specified:

*	The ASPs that are currently part of the thread's library name space will be searched to locate the library. This includes the system ASP (ASP 1), all defined basic user ASPs (ASPs 2-32), and, if the thread has an ASP group, the primary and secondary ASPs in the thread's ASP group.
*SYSBAS	The system ASP (ASP 1) and all defined basic user ASPs (ASPs 2-32) will be searched to locate the library. No primary or secondary ASPs will be searched, even if the thread has an ASP group.
*CURASPGRP	If the thread has an ASP group, the primary and secondary ASPs in the ASP group will be searched to locate the library. The system ASP (ASP 1) and defined basic user ASPs (ASPs 2-32) will not be searched.
*ALLAVL	All available ASPs will be searched. This includes the system ASP (ASP 1), all defined basic user ASPs (ASPs 2-32), and all available primary and secondary ASPs (ASPs 33-255 with a status of 'Available'). The ASP groups are searched in alphabetical order by the primary ASP. The system ASP and all defined basic user ASPs are searched after the ASP groups. ASPs and libraries to which the user is not authorized are bypassed and no authority error messages are sent. The search ends when the first object is found of the specified object name, library name and object type. If the user is not authorized to the object, an authority error message is sent.

Auxiliary storage pool (ASP) search type. The type of the search when a specific auxiliary storage pool (ASP) device name is specified for the ASP device name field. This field must be blanks when a special value is specified for the auxiliary storage pool (ASP) device name field. One of the following values may be specified:

*ASP	Only the single ASP named in the auxiliary storage pool (ASP) device name field will be searched.
*ASPGRP	All ASPs in the auxiliary storage pool (ASP) group named in the auxiliary storage pool (ASP) device name field will be searched. The device name must be the name of the primary auxiliary storage pool (ASP) in the group.

Length of auxiliary storage pool (ASP) control format. The total length of the auxiliary storage pool (ASP) control format. The length can be 0 bytes to indicate that no auxiliary storage pool (ASP) control information is provided. Otherwise, the length must be 24 bytes. An error is returned if the length specified is not 24 or 0.

OBJD0100 Format

The following information is returned for the OBJD0100 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	CHAR(10)	Object name
18	12	CHAR(10)	Object library name
28	1C	CHAR(10)	Object type
38	26	CHAR(10)	Return library
48	30	BINARY(4)	» Object ASP number«
52	34	CHAR(10)	Object owner
62	3E	CHAR(2)	Object domain
64	40	CHAR(13)	Creation date and time
77	4D	CHAR(13)	Object change date and time

OBJD0200 Format

The following information is returned for the OBJD0200 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0		Everything from the OBJD0100 format
90	5A	CHAR(10)	Extended object attribute
100	64	CHAR(50)	Text description
150	96	CHAR(10)	Source file name
160	A0	CHAR(10)	Source file library name
170	AA	CHAR(10)	Source file member name

OBJD0300 Format

The following information is returned for the OBJD0300 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	Hex		
0	0		Everything from the OBJD0200 format
180	B4	CHAR(13)	Source file updated date and time

193	C1	CHAR(13)	Object saved date and time
206	CE	CHAR(13)	Object restored date and time
219	DB	CHAR(10)	Creator's user profile
229	E5	CHAR(8)	System where object was created
237	ED	CHAR(7)	Reset date
244	F4	BINARY(4)	Save size
248	F8	BINARY(4)	Save sequence number
252	FC	CHAR(10)	Storage
262	106	CHAR(10)	Save command
272	110	CHAR(71)	Save volume ID
343	157	CHAR(10)	Save device
353	161	CHAR(10)	Save file name
363	16B	CHAR(10)	Save file library name
373	175	CHAR(17)	Save label
390	186	CHAR(9)	System level
399	18F	CHAR(16)	Compiler
415	19F	CHAR(8)	Object level
423	1A7	CHAR(1)	User changed
424	1A8	CHAR(16)	Licensed program
440	1B8	CHAR(10)	Program temporary fix (PTF)
450	1C2	CHAR(10)	Authorized program analysis report (APAR)

OBJD0400 Format

The following information is returned for the OBJD0400 format. For detailed descriptions of the fields in the table, see Field Descriptions.

Offset		Type	Field
Dec	**Hex**		
0	0		Everything from the OBJD0300 format
460	1CC	CHAR(7)	Last-used date
467	1D3	CHAR(1)	Usage information updated
468	1D4	BINARY(4)	Days-used count
472	1D8	BINARY(4)	Object size
476	1DC	BINARY(4)	Object size multiplier
480	1E0	CHAR(1)	Object compression status
481	1E1	CHAR(1)	Allow change by program
482	1E2	CHAR(1)	Changed by program
483	1E3	CHAR(10)	User-defined attribute
493	1ED	CHAR(1)	Object overflowed ASP indicator
494	1EE	CHAR(13)	Save active date and time
507	1FB	CHAR(10)	Object auditing value

517	205	CHAR(10)	Primary group
527	20F	CHAR(1)	Journal status
528	210	CHAR(10)	Journal name
538	21A	CHAR(10)	Journal library name
548	224	CHAR(1)	Journal images
549	225	CHAR(1)	Journal entries to be omitted
550	226	CHAR(13)	Journal start date and time
563	233	CHAR(1)	Digitally signed
564	234	BINARY(4)	Saved size in units
568	238	BINARY(4)	Saved size multiplier
572	23C	BINARY(4)	Library ASP number
576	240	CHAR(10)	Object ASP device name
586	24A	CHAR(10)	Library ASP device name
596	254	CHAR(1)	Digitally signed by system-trusted source
597	255	CHAR(1)	Digitally signed more than once

Field Descriptions

Allow change by program. A 1-character variable that is used to return the allow change by program flag. A 1 is returned if the object can be changed with the Change Object Description (QLICOBJD) API. A 0 is returned if the object cannot be changed with the API.

Authorized program analysis report (APAR). The identifier of the authorized program analysis report (APAR) that caused this object to be replaced. The field is blank if the object did not change because of an APAR.

Bytes available. The length of all data available to return. All available data is returned if enough space is provided.

Bytes returned. The length of the data actually returned.

Changed by program. A 1-character variable that is used to return the changed by program flag. A 1 is returned if the object has been changed with the QLICOBJD API. A 0 is returned if the object has not been changed by the API.

Compiler. The licensed program identifier, version number, release level, and modification level of the compiler.

The field has a ppppppVvvRrrMmm format where:

ppppppp	The licensed program identifier.
Vvv	The character V is followed by a 2-character version number.
Rrr	The character R is followed by a 2-character release level.
Mmm	The character M is followed by a 2-character modification level.

The field is blank if you do not compile the program.

Creation date and time. The date and time the object was created. The creation date and time field is in the CYYMMDDHHMMSS format:

C	Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
YY	Year
MM	Month
DD	Day
HH	Hour
MM	Minute
SS	Second

Creator's user profile. The name of the user that created the object.

Days-used count. The number of days the object was used. If the object does not have a last used date, the count is 0.

Digitally signed. A 1-character variable that indicates whether the object »has an OS/400 digital signature.

0	The object does not have an OS/400 digital signature.
1	The object has an OS/400 digital signature.«

»**Digitally signed by system-trusted source.** A 1-character variable that indicates whether the object is signed by a source that is trusted by the system.

0	None of the object signatures came from a source that is trusted by the system.
1	The object is signed by a source that is trusted by the system. If the object has multiple signatures, at least one of the signatures came from a source that is trusted by the system.

Digitally signed more than once. A 1-character variable that indicates whether the object has more than one OS/400 digital signature.

0	The object has only one OS/400 digital signature or does not have an OS/400 digital signature. Refer to the digitally signed variable to determine whether the object has an OS/400 digital signature.
1	The object has more than one OS/400 digital signature. Refer to the digitally signed by system-trusted source variable to determine whether the object has an OS/400 digital signature from a source trusted by the system.«

Extended object attribute. The extended attribute of the object, such as a program or file type. Extended attributes further describe the object. For example, an object type of *PGM may have a value of RPG (RPG program) or CLP (CL program), and an object type of *FILE may have a value of PF (physical file), LF (logical file), DSPF (display file), SAVF (save file), and so on.

Journal entries to be omitted. The journal entries to be omitted. The field is 1 if *open* and *close* operations do not generate *open* and *close* journal entries. The field is 0 if no entries are omitted. This field is blank if the object has never been journaled.

Journal images. The type of images that are written to the journal receiver for updates to the object. The field is 0 if only *after* images are generated for changes to the object. The field is 1 if both *before* and *after* images are generated for changes to the object. This field is blank if the object has never been journaled.

Journal library name. The name of the library containing the journal. This field is blank if the object has never been journaled.

Journal name. The name of the current or last journal. This field is blank if the object has never been journaled.

Journal start date and time. The time at which journaling for the object was last started. The format is the same as the creation date description. This field is blank if the object has never been journaled.

Journal status. The 1-character variable that returns the current journaling status of an object. The value is 1 if the object is currently being journaled; the value is 0 if the object is currently not being journaled.

Last-used date. The date the object was last used. This field is in the CYYMMDD format, which is the same format used for the reset date. If the object has no last-used date, the field is blank.

Licensed program. The name, release level, and modification level of the licensed program if the retrieved object is part of a licensed program. The 7-character name starts in character position 1, the version number starts in position 8, the release level starts in position 11, and the modification level starts in position 14. The field is blank if the retrieved object is not a part of a licensed program.

»**Library ASP device name.** The name of the auxiliary storage pool (ASP) device where storage is allocated for the library containing the object. The following special values can be returned:

**N*	The name of the ASP device cannot be determined.
**SYSBAS*	System ASP (ASP 1) or defined basic user ASPs (ASPs 2-32)

Library ASP number. The number of the auxiliary storage pool (ASP) where storage is allocated for the library containing the object. A value from one of the following ranges is returned:

1	System ASP
2-32	Basic user ASP
33-255	Primary or secondary ASP«

Object auditing value. A 10-character variable that is used to return the type of auditing for an object. The valid values are:

**NONE*	No auditing occurs for this object when it is read or changed regardless of the user who is accessing the object.
**USRPRF*	Audit this object only if the » current user is being audited. The current user« is tested to determine if auditing should be done for this object. The user profile can specify if only change access is audited or if both read and change accesses are audited for this object.
**CHANGE*	Audit all change access to this object by all users on the system.
**ALL*	Audit all access to this object by all users on the system. All access is defined as a read or change operation.

Object change date and time. The date and time the object was last changed. The format is the same as the creation date description, or it is blank if the object was not changed.

Object compression status. Whether the object is compressed or decompressed. The status is returned in a 1-character variable with one of these values:

Y Compressed.

N Permanently decompressed and compressible.

X Permanently decompressed and *not* compressible.

T Temporarily decompressed.

F Saved with storage freed; compression status cannot be determined.

Temporarily decompressed objects exist in both decompressed and compressed form. *Permanently* decompressed objects exist in decompressed form only. The system handles some decompression automatically, depending on the type of object, the operation performed on it, and its frequency of use. For an overview of object compression and decompression, see the CL Programming book. For details about how to explicitly compress and decompress objects, see the online help for these commands: Compress Object (CPROBJ), Decompress Object (DCPOBJ), and Reclaim Temporary Storage (RCLTMPSTG).

Object domain. The domain that contains the object. The value is *U if the object is in the user domain, or *S if the object is in the system domain.

Object level. The object control level for the created object.

»**Object ASP device name.** The name of the auxiliary storage pool (ASP) device where storage is allocated for the object. The following special values can be returned:

**N* The name of the ASP device cannot be determined.

**SYSBAS* System ASP (ASP 1) or defined basic user ASPs (ASPs 2-32)

Object ASP number. The number of the auxiliary storage pool (ASP) where storage is allocated for the object. A value from one of the following ranges is returned:

1 System ASP

2-32 Basic user ASP

33-255 Primary or secondary ASP«

Object library name. The name of the library containing the object.

Object name. The name of the object.

Object overflowed ASP indicator. The 1-character variable that returns the object overflowed auxiliary storage pool (ASP) indicator. The value is 1 if the object overflowed the ASP in which it resides; the value is 0 if the object has not overflowed the ASP. For objects in the system ASP» (ASP 1) or in a primary or secondary ASP (ASPs 33-255), a 0 is always returned because an object that resides in the system ASP or in a primary or secondary ASP «cannot overflow its ASP.

Object owner. The name of the object owner's user profile.

Object restored date and time. The date and time the object was last restored. The format is the same as for the creation date, or it is blank if the object was never restored.

Object saved date and time. The date and time the object was last saved. The format is the same as for the creation date description, or it is blank if the object was never saved.

Object size. The size of the object in units of the size multiplier. The object size is equal to or smaller than the object size multiplied by the object size multiplier.

Object size multiplier. The value to multiply the object size by to get the true size. The value is 1 if the object is smaller than 1 000 000 000 bytes, and 1024 if it is larger.

Object type. The object type. For a list of all the available o bject types, see the Control Language (CL) information in the iSeries Information Center.

Primary group. The name of the user who is the primary group for the object. If no primary group exists for the object, this field contains a value of *NONE.

Program temporary fix (PTF). The number of the program temporary fix (PTF) number that caused this object to be replaced. This field is blank if the object was not changed because of a PTF.

Reset date. The date the days-used count was last reset to 0. The reset date field is in the CYYMMDD format:

C	Century, where 0 indicates years 19*xx* and 1 indicates years 20*xx*.
YY	Year
MM	Month
DD	Day

If the days-used count was not reset, the date is blank.

Return library. The name of the library containing the object if *LIBL or *CURLIB is specified for the library name on the object parameter.

Save active date and time. The date and time the object was last saved when the SAVACT(*LIB, *SYSDFN, or *YES) save operation was specified, in system time-stamp format. This parameter is found on the Save Library (SAVLIB), Save Object (SAVOBJ), Save Changed Object (SAVCHGOBJ), and Save Document Library Object (SAVDLO) CL commands. The format is the same as for the creation date description, or it is blank if the object was never saved or if SAVACT(*NO) was specified on the last save operation for the object.

Save command. The command used to save the object. The field is blank if the object was not saved.

Save device. The type of device to which the object was last saved. The field is *SAVF if the last save operation was to a save file. The field is *DKT if the last save operation was to diskette. The field is *TAP if the last save operation was to tape. The field is *OPT if the last save operation was to optical. The field is blank if the object was not saved.

Save file library name. The name of the library that contains the save file if the object was saved to a save file. The field is blank if the object was not saved to a save file.

Save file name. The name of the save file if the object was saved to a save file. The field is blank if the object was not saved to a save file.

Save label. The file label used when the object was saved. The variable is blank if the object was not saved to tape, diskette, or optical. The value of the variable corresponds to the value specified for the LABEL or OPTFILE parameter on the command used to save the object.

Save sequence number. The tape sequence number assigned when the object was saved on tape. If the object was not saved to tape, the field contains zeros.

Save size. The size of the object in bytes of storage at the time of the last save operation. The field contains zeros if the object was not saved. This field will contain a size up to 2 GB. If the save size is actually greater than 2GB, -1 is returned in this field. Fields save size in units and save size multiplier should be used to get the save size that is larger than 2GB.

Save size in units. The size of the object in units of the size multiplier at the time of the last save operation. The save size is equal to or smaller than the save size multiplied by the save size multiplier. The field contains zeros if the object was not saved.

Save size multiplier. The value to multiply the save size (in units) by to get the true size. The value is 1 if the save size is smaller than 1 000 000 000 bytes, and 1024 if it is larger.

Save volume ID. The tape, diskette, or optical volumes that are used for saving the object. The variable returns a maximum of 10 six-character volumes. The volume IDs begin in character positions 1, 8, 15, 22, 29, 36, 43, 50, 57, and 64. Each volume ID entry is separated by a single character. If the object was saved in parallel format, the separator character contains a 2 before the first volume in the second media file, a 3 before the third media file, and so on, up to a 0 before the tenth media file. Otherwise, the separator characters are blank. If more than 10 volumes are used and the object was saved in serial format, 1 is returned in the 71st character of the variable. If the object was saved in parallel format, a 2 is returned in the 71st character of the variable. Otherwise, the 71st character is blank. The field is blank if the object was last saved to a save file or if it was never saved.

Source file library name. The name of the library that contains the source file used to create the object. The field is blank if no source file created the object.

Source file member name. The name of the member in the source file. The field is blank if no source file created the object.

Source file name. The name of the source file used to create the object. The field is blank if no source file created the object.

Source file updated date and time. The date and time the member in the source file was last updated. The field is in the same format as the creation time and date. The field is blank if no source file created the object.

Storage. The storage status of the object data. *FREE indicates the object data is freed and the object is suspended. *KEEP indicates the object data is not freed and the object is not suspended.

System level. The level of the operating system when the object was created.

The field has a VvvRrrMmm format where:

Vvv	The character V is followed by a 2-character version number.
Rrr	The character R is followed by a 2-character release level.
Mmm	The character M is followed by a 2-character modification level.

System where object was created. The name of the system on which the object was created.

Text description. The text description of the object. The field is blank if no text description is specified.

Usage information updated. Whether the object usage information is updated for this object type. The indicator is returned as Y (Yes) or N (No).

User changed. Whether the user program was changed. A character 1 is returned if the user changed the object. If the object was not changed by the user, the field is character 0.

User-defined attribute. Further defines an object type. This field is set by the user while using the QLICOBJD API.

Error Messages

Message ID	Error Message Text
» CPFB8ED E	Device description &1 not correct for operation.«
CPF21AC E	Length or displacement value &1 not valid.
CPF2101 E	Object type *&1 not valid.
CPF2115 E	Object &1 in &2 type *&3 damaged.
CPF2150 E	Object information function failed.
CPF2151 E	Operation failed for &2 in &1 type *&3.
» CPF2173 E	Value for ASPDEV not valid with special value for library.«
» CPF218C E	&1 not a primary or secondary ASP.«
» CPF218D E	&1 not a primary ASP when *ASPGRP specified.«
CPF2451 E	Message queue &1 is allocated to another job.
CPF3CF1 E	Error code parameter not valid.
CPF3C07 E	Error occurred while retrieving information from object &1.
CPD3C20 D	Error occurred with receiver variable specified.
CPD3C21 D	Format name &1 is not valid.
CPD3C24 D	Length of the receiver variable is not valid.
CPD3C31 D	Object type &1 is not valid.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
» CPF3C3B E	Value for parameter &2 for API &1 not valid.«
CPF3C36 E	Number of parameters, &1, entered for this API was not valid.
CPF3C90 E	Literal value cannot be changed.
» CPF3202 E	File &1 in library &2 in use.«
» CPF3203 E	Cannot allocate object for file &1 in &2.«

CPF36F7 E	Message queue QSYSOPR is allocated to another job.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
» CPF980B E	Object &1 in library &2 not available.«
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.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9811 E	Program &1 in library &2 not found.
CPF9812 E	File &1 in library &2 not found.
CPF9814 E	Device &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9821 E	Not authorized to program &1 in library &2.
CPF9822 E	Not authorized to file &1 in library &2.
CPF9825 E	Not authorized to device &1.
CPF9830 E	Cannot assign library &1.
CPF9831 E	Cannot assign device &1.
» CPF9833 E	*CURASPGRP or *ASPGRPPRI specified and thread has no ASP group.«
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API Introduced: V1R3

Top | Object APIs | APIs by category