

UNIX-Type APIs (V5R2)

Resource Reservation Setup Protocol APIs

Table of Contents

[Resource Reservation Setup Protocol APIs](#)

- [qtoq_accept\(\)](#) (Accept QoS Sockets Connection)❧
- [qtoq_close\(\)](#) (Close QoS Sockets Connection)❧
- [qtoq_connect\(\)](#) (Make QoS Sockets Connection)❧
- [qtoq_ioctl\(\)](#) (Set QoS Sockets Control Options)❧
- [QgyOpenListQoSMonitorData](#) (Open List of QoS Monitor Data)
- [QtoqDeleteQoSMonitorData](#) (Delete QoS Monitor Data)❧
- [QtoqEndQoSMonitor](#) (End QoS Monitor)
- [QtoqListSavedQoSMonitorData](#) (List Saved QoS Monitor Data)❧
- [QtoqSaveQoSMonitorData](#) (Save QoS Monitor Data)❧
- [QtoqStartQoSMonitor](#) (Start QoS Monitor)
- [rapi_dispatch\(\)](#) (Dispatch the RAPI message handling routine defined in the rapi_session() call)
- [rapi_fmt_adspec\(\)](#) (Format a RAPI Adspec into a string suitable for printing)
- [rapi_fmt_filtspec\(\)](#) (Format a RAPI Filter spec into a string suitable for printing)
- [rapi_fmt_flowspec\(\)](#) (Format a RAPI Flowspec into a string suitable for printing)
- [rapi_fmt_tspec\(\)](#) (Format a RAPI Tspec into a string suitable for printing)
- [rapi_getfd\(\)](#) (Get a descriptor to wait on before dispatching the RAPI message handling routine)
- [rapi_release\(\)](#) (Release the currently active RAPI reservation and close the open sessions)
- [rapi_reserve\(\)](#) (Make, modify, or delete a RAPI reservation)
- [rapi_sender\(\)](#) (Identify a RAPI sender)
- [rapi_session\(\)](#) (Create a RAPI session)
- [rapi_version\(\)](#) (Retrieve the current RAPI version)

[Header Files for UNIX-Type Functions](#)

[Errno Values for UNIX-Type Functions](#)

Resource Reservation Setup Protocol APIs

The resource reservation protocol (RSVP), along with the RAPI APIs, perform your integrated services reservation. This protocol is part of the Quality of service (QoS) function that allows you to request network priority and bandwidth for TCP/IP applications. The RSVP protocol is used to load rules to the TCP/IP stack that controls these requests. These rules are called IntServ rules. QoS also allows the user to define DiffServ rules that request special handling in the network for groups of applications or connections. See [Quality of service \(QoS\)](#) for more information.

The six monitor APIs can be used to retrieve information on both IntServ and DiffServ rules.

Note: A thorough understanding of the RSVP protocol and the contents of Internet RFC 2205 is required to be able to use the RAPI APIs correctly. These APIs will not function unless the proper sequencing of events between the client and server is observed.

The Resource Reservation Setup Protocol APIs are:

- [»qtoq_accept\(\)](#) (Accept QoS Sockets Connection) provides simplified Quality of Service support for connection-oriented sockets communications between RSVP aware applications on a client and server.◀◀
- [»qtoq_close\(\)](#) (Close QoS Sockets Connection) is called to close the socket and QoS session that was created using the other qtoq_ sockets-type APIs.◀◀
- [»qtoq_connect\(\)](#) (Make QoS Sockets Connection) provides simplified Quality of Service functionality for connection-oriented sockets communications between RSVP aware applications on a client and server.◀◀
- [»qtoq_ioctl\(\)](#) (Set QoS Sockets Control Options) provides simplified Quality of Service functionality for connectionless sockets communications between RSVP aware applications on a client and server.◀◀
- [QgyOpenListQoSMonitorData](#) (Open List of QoS Monitor Data) allows the user to gathering information related to QoS services.
- [»QtoqDeleteQoSMonitorData](#) (Delete QoS Monitor Data) allows the user to delete a particular and/or a group of collected QoS monitor data.◀◀
- [QtoqEndQoSMonitor](#) (End QoS Monitor) allows the user to stop gathering information related to QoS services.
- [»QtoqListSavedQoSMonitorData](#) (List Saved QoS Monitor Data) allows the user to return a list of all collected monitor data that was saved previously.◀◀
- [»QtoqSaveQoSMonitorData](#) (Save QoS Monitor Data) allows the user to save a copy of the collected QoS monitor data for future use.◀◀
- [QtoqStartQoSMonitor](#) (Start QoS Monitor) allows the user to gathering information related to QoS services.
- [rapi_dispatch\(\)](#) (Dispatch the RAPI message handling routine defined in the rapi_session() call) dispatches the RAPI message-handling routine defined in the rapi_session() call.
- [rapi_fmt_adspec\(\)](#) (Format a RAPI Adspec into a string suitable for printing) formats a RAPI Adspec into a string suitable for printing by converting the RAPI Adspec information that has been passed to the API into a string in the supplied buffer.
- [rapi_fmt_filtspec\(\)](#) (Format a RAPI Filter spec into a string suitable for printing) formats a RAPI Filter spec into a string suitable for printing by converting the RAPI filtspec information that has been passed to the API into a string in the buffer that has been passed to the API.

- [rapi_fmt_flowspec\(\)](#) (Format a RAPI Flowspec into a string suitable for printing) formats a RAPI Flowspec into a string suitable for printing by converting the RAPI flowspec information that has been passed to the API into a character string in the buffer that was passed to the API.
- [rapi_fmt_tspec\(\)](#) (Format a RAPI Tspec into a string suitable for printing) formats a RAPI Tspec into a string suitable for printing by converting the RAPI Tspec information that has been passed to the API into a string in the buffer that has been passed to the API.
- [rapi_getfd\(\)](#) (Get a descriptor to wait on before dispatching the RAPI message handling routine) returns the file descriptor associated with a successful rapi_session() call.
- [rapi_release\(\)](#) (Release the currently active RAPI reservation and close the open sessions) releases the RAPI reservation that is active currently and closes the open sessions.
- [rapi_reserve\(\)](#) (Make, modify, or delete a RAPI reservation) used to make, modify, or delete an RSVP reservation in the network.
- [rapi_sender\(\)](#) (Identify a RAPI sender) identifies an RSVP sender to potential receivers of the data.
- [rapi_session\(\)](#) (Create a RAPI session) returns an API session ID that is unique to this request.
- [rapi_version\(\)](#) (Retrieve the current RAPI version) returns the RAPI version currently being used by the RSVP agent.

»qtoq_accept()--Accept QoS Sockets Connection API

Syntax

```
#include <qtoqsapi.h>

int qtoq_accept(
    int                socket_descriptor,
    int                req_type,
    struct sockaddr    *address,
    int                *address_length,
    qos_req            *qos_data,
    unsigned int       *qos_session,
    int                *qos_descriptor,
    )
```

Service Program Name: QSYS/QTOQSAPI

Default Public Authority: *EXCLUDE

Threadsafe: Yes

The **qtoq_accept()** API provides simplified Quality of Service support for connection-oriented sockets communications between RSVP aware applications on a client and server. The standard **accept()** sockets call can be replaced with this API.

Parameters

socket_descriptor

(Input) Required

An opened socket descriptor that has been bound to the IP address and port from which the application will accept connection requests.

req_type

Input) Required

The type of QoS service being requested. The possible values are:

- | | |
|------------------------------------|---|
| <i>REQ_SIGNAL_RET_EVENTS</i> (1) | Use normal RSVP signaling and return RSVP events to the calling program. |
| <i>REQ_SIGNAL_NORET_EVENTS</i> (2) | Use normal RSVP signaling without returning events to the calling program. |
| <i>REQ_NOSIGNAL</i> (3) | See if the RSVP rule for the requested connection has been defined as "no signaling." If yes, then load the requested rule. |

address

(Output) Required

Pointer to a sockaddr structure where the IP address and port of the client requesting the connection will be stored.

Address_length

(Input/Output) Required

Pointer to an integer where the size of the address variable is given to the API and the length of the returned client address will be stored.

qos_data

(Input) Required

Pointer to a qos_req data structure that defines the type of service being requested and the source and destination addresses of the request.

The qos_req data structure is defined as follows:

```
typedef struct
{
    int          service;      /* Values can be GUARANTEED_SERV (2)
                               or CONTROLLED_LOAD_SERV (5) */
    union
    {
        struct CL_spec        /* Controlled-Load service */
        {
            float      TB_Tspec_r;    /* token bucket rate in bytes/sec */
            float      TB_Tspec_b;    /* token bucket depth in bytes */
            float      TB_Tspec_p;    /* token bucket peak in bytes/sec */
            unsigned long TB_Tspec_m;  /* min policed unit in bytes */
            unsigned long TB_Tspec_M;  /* max packet size in bytes */
        } CL_spec;
        struct Guar_spec      /* Guaranteed service */
        {
            float      Guar_R;        /* guaranteed rate in bytes/sec */
            unsigned long Guar_S;     /* slack term in microseconds */
        } Guar_spec;
    } spec_u;
} qos_spec_t;

typedef struct
{
    struct sockaddr    dest;          /* Destination address/port */
    int                d_length;     /* Destination address length */
    struct sockaddr    source;       /* Source address/port */
    int                s_length;     /* Source address length */
    int                style;        /* Style of Reservation. */
    qos_spec_t         Spec;         /* Flow info */
    unsigned char      result;       /* API status */
} qos_req; /* End of QoS request structure */
```

qos_session

(Output) Required

Pointer to an integer value where the unique QoS session ID can be stored. This ID is required for all future QoS API calls.

qos_descriptor

(Output) Optional

Pointer to an integer where the value of the descriptor that the application can wait on for RSVP events is

stored. This value is set to NULL if it is not used.

Authorities

None.

Return Values

0 if successful.

-1 if function failed. Errno indicates error reason.

Error Conditions

When `qtoq_accept()` fails `errno` can be set to one of the following:

[EBADF]

Descriptor not valid.

[EFAULT]

Bad address.

[ECONNABORTED]

Connection ended abnormally. An `accept()` was issued on a socket for which receives have been disallowed (due to a `shutdown()` call).

This also could be encountered if time elapsed since a successful `Rbind()` is greater than the margin allowed by the associated SOCKS server.

[EFAULT]

Bad address. System detected an address that was not valid while attempting to access the address or `address_length` parameters.

[EINTR]

Interrupted function call.

[EINVAL]

Parameter not valid. This error code indicates one of the following:

- The `address_length` parameter is set to a value that is less than zero, and the `address` parameter is set to a value other than a NULL pointer.
- A `listen()` has not been issued against the socket referenced by the `socket_descriptor` parameter.

[EIO]

Input/output error.

[EMFILE]

Too many descriptions for this process.

[ENFILE]

Too many descriptions in system.

[ENOBUFFS]

There is not enough buffer space for the requested operation.

[ENOTSOCK]

The specified descriptor does not reference a socket.

[EOPNOTSUPP]

Operation not supported. The socket_descriptor parameter references a socket that does not support the accept(). The accept() is valid only on sockets that are connection-oriented (for example, type of SOCK_STREAM).

[EUNATCH]

The protocol required to support the specified address family is not available at this time.

[EUNKNOWN]

Unknown system state.

[EWOULDBLOCK]

Operation would have caused the thread to be suspended.

Error Messages

| Message ID | Error Message Text |
|------------|--|
| CPE3418 E | Possible APAR condition or hardware failure. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |
| CPFA081 E | Unable to set return value or error code. |

Usage Notes

1. The application program can choose to be signaled when RSVP events occur or allow the QoS server to handle the events. If the server handles the events, the application program will not be informed if the RSVP signaling fails or if the requested reservations have been changed by the network.
2. The REQ_NOSIGNAL request type will be honored only if a policy exists that matches the requested connection and it is marked as a "no signaling" policy. Otherwise, an *[ENOTSUPPORT]* error will be returned.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#).



API Introduced: V5R2

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

>qtoq_close()--Close QoS Sockets Connection API

Syntax

```
#include <qtoqsapi.h>

int qtoq_close(
    int                socket_descriptor,
    int                *qos_descriptor,
    unsigned int       *qos_session,
    )
```

Service Program Name: QSYS/QTOQSAPI

Default Public Authority: *EXCLUDE

Threadsafe: Yes

qtoq_close() is called to close the socket and QoS session that was created using the other qtoq_sockets-type APIs. It performs a standard sockets close(); on the socket descriptor, close the QoS session for this connection and inform the QoS server that the connection should be closed and the rule unloaded.

Parameters

socket_descriptor

(Input) Required

The socket descriptor that was created to perform the TCP/IP communications for this connection.

qos_descriptor

(Input) Optional

Pointer to an integer for the value of the descriptor that the application used to wait on QoS events.

qos_session

(Input) Required

Pointer to an integer containing the QoS session ID that was returned when the QoS connection was established.

Authorities

None.

Return Values

0 if successful.

-1 if function failed. Errno indicates error reason.

Error Conditions

When this function call fails, the errno value is set to one of the following:

[EBADF]

Descriptor not valid.

[EIO]

Input/output error.

[ENOBUFS]

There is not enough buffer space for the requested operation.

[EUNKNOWN]

Unknown system state.


Error Messages

| Message ID | Error Message Text |
|------------|--|
| CPE3418 E | Possible APAR condition or hardware failure. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |
| CPFA081 E | Unable to set return value or error code. |

Usage Notes

1. The `qtoq_close()` API must be used in place of the normal `close()` sockets call in an application using the QTOQ APIs. If it is not used, the results are unpredictable.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#). 



API Introduced: V5R2

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

»qtoq_connect()--Make QoS Sockets Connection API

Syntax

```
#include <qtoqsapi.h>

int qtoq_connect(
    int                socket_descriptor,
    struct sockaddr    *address,
    int                address_length,
    int                req_type,
    qos_conn_req      *qos_data,
    unsigned int       *qos_session,
    int                *qos_descriptor,
    )
```

Service Program Name: QSYS/QTOQSAPI

Default Public Authority: *EXCLUDE

Threadsafe: Yes

The **qtoq_connect()** API provides simplified Quality of Service functionality for connection-oriented sockets communications between RSVP aware applications on a client and server. The standard **connect()** sockets call can be replaced with this API.

Parameters

socket_descriptor

(Input) Required

An opened socket descriptor that has been bound to the IP address and port from which the application will accept connection requests.

destination_address

(Input) Required

A pointer to a sockaddr structure containing the IP address and port of the server to connect to.

address_length

(Input) Required

Integer containing the length of the destination address structure.

req_type

(Input) Required

The type of QoS service being requested. The possible values are:

| | |
|---------------------------------|--|
| <i>REQ_SIGNAL_RET_EVENTS(1)</i> | Use normal RSVP signaling and return RSVP events to the calling program. |
|---------------------------------|--|

REQ_SIGNAL_NORET_EVENTS(2) Use normal RSVP signaling without returning events to the calling program.

qos_data

(Input) Required

Pointer to a *qos_conn_req* data structure that defines the type of service being requested and the source and destination addresses of the request.

The *qos_conn_req* data structure is defined below:

```
typedef struct
{
    int          service; ;      /* Values can be GUARANTEED_SERV (2)
                                or CONTROLLED_LOAD_SERV (5) */
    union
    {
        struct  CL_spec          /* Controlled-Load service          */
        {
            float      TB_Tspec_r; /* token bucket rate in bytes/sec */
            float      TB_Tspec_b; /* token bucket depth in bytes     */
            float      TB_Tspec_p; /* token bucket peak in bytes/sec  */
            unsigned long TB_Tspec_m; /* min policed unit in bytes      */
            unsigned long TB_Tspec_M; /* max packet size in bytes       */
        } CL_spec;
        struct Guar_spec /* Guaranteed service          */
        {
            float      Guar_R; /* guaranteed rate in bytes/sec    */
            unsigned long Guar_S; /* slack term in microseconds     */
        } Guar_spec;
    } spec_u;
} qos_spec_t;

typedef struct
{
    struct sockaddr      source; /* Source address/port          */
    int                  s_length; /* Source address length        */
    int                  style; /* Style of Reservation.        */
    qos_spec_t           Spec; /* Flow info                     */
    unsigned char        result; /* API status                    */
} qos_conn_req; /* End of QoS connection request structure */
```

qos_session

(Output) Required

Pointer to an integer value where the unique QoS session ID can be stored. This ID is required for all future QoS API calls.

qos_descriptor

(Output) Optional

Pointer to an integer where the value of the descriptor that the application can wait on for RSVP events is stored. This value is set to NULL if it is not used.

Authorities

None.

Return Values

- 0* if successful.
- 1* if function failed. Errno indicates error reason.

Error Conditions

When this function call fails, the errno value is set to one of the following:

[EACCES]

Permission denied. This error code indicates one of the following:

- The process does not have the appropriate privileges to connect to the address pointed to by the *destination_address* parameter.
- The socket pointed to by *socket_descriptor* is using a connection-oriented transport service, and the *destination_address* parameter specifies a TCP/IP limited broadcast address (internet address of all ones).

[EADDRINUSE]

Address already in use. This error code indicates one of the following:

- The *socket_descriptor* parameter points to a connection-oriented socket that has been bound to a local address that contained no wildcard values, and the *destination_address* parameter specified an address that matched the bound address.
- The *socket_descriptor* parameter points to a socket that has been bound to a local address that contained no wildcard values, and the *destination_address* parameter (also containing no wildcard values) specified an address that would have resulted in a connection with an association that is not unique.

[EADDRNOTAVAIL]

Address not available. This error code indicates one of the following:

- The *socket_descriptor* parameter points to a socket with an address family of AF_INET and either a port was not available or a route to the address specified by the *destination_address* parameter could not be found.

[EAFNOSUPPORT]

The type of socket is not supported in this protocol family. The address family specified in the address structure pointed to by the *destination_address* parameter cannot be used with the socket pointed to by the *socket_descriptor* parameter. This error also will be reported if the API is called with a socket type that is not AF_INET and SOCK_DGRAM or SOCK_STREAM.

[EALREADY]

Operation already in progress. A previous connect() function had already been issued for the socket pointed to by the socket_descriptor parameter, and has yet to be completed. This error code is returned only on sockets that use a connection-oriented transport service.

[EBADF]

Descriptor not valid.

[ECONNREFUSED]

The destination socket refused an attempted connect operation. This error occurs when there is no application that is bound to the address specified by the destination_address parameter.

[EFAULT]

Bad address. The system detected an address that was not valid while attempting to access the destination_address parameter.

[EHOSTUNREACH]

A route to the remote host is not available.

[EINPROGRESS]

Operation in progress. The socket_descriptor parameter points to a socket that is marked as non blocking and the connection could not be completed immediately. This error code is returned only on sockets that use a connection-oriented transport service.

[EINTR]

Interrupted function call.

[EINVAL]

Parameter not valid. This error code indicates one of the following:

- The address_length parameter specifies a length that is negative or not valid for the address family.
- The AF_INET socket is of type SOCK_STREAM, and a previous connect() has already completed unsuccessfully. Only one connection attempt is allowed on a connection-oriented socket.

[EIO]

Input/output error.

[EISCONN]

A connection has already been established. This error code is returned only on sockets that use a connection-oriented transport service.

[ENETUNREACH]

Cannot reach the destination network. This error code indicates the following:

- For sockets that use the AF_INET address family, the address specified by the destination_address parameter requires the use of a router, and the socket option SO_DONTROUTE is currently set on.

[ENOBUFS]

There is not enough buffer space for the requested operation.

[ENOTDIR]

Not a directory.

[EOPNOTSUPP]

Operation not supported.

[ETIMEDOUT]

A remote host did not respond within the timeout period. This error code is returned when connection establishment times out. No connection is established. A possible cause may be that the partner application is bound to the address specified by the `destination_address` parameter, but the partner application has not yet issued a `listen()`.

[EUNKNOWN]

Unknown system state.

[EUNATCH]

The protocol required to support the specified address family is not available at this time.

[EPROTO]

An underlying protocol error has occurred.

Error Messages

| Message ID | Error Message Text |
|-------------------|--|
| CPE3418 E | Possible APAR condition or hardware failure. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |
| CPFA081 E | Unable to set return value or error code. |

Usage Notes

1. The `qtoq_connect()` API can be used to replace the normal `connect()` sockets call in an application using connection oriented sockets.
2. The application program can choose to be signaled when RSVP events occur or allow the QoS server to handle the events. If the server handles the events, the application program will not be informed if the RSVP signaling fails or if the requested reservations have been changed by the network.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#).



API Introduced: V5R2

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

qtoq_ioctl()--Set QoS Sockets Control Options API

Syntax

```
#include <qtoqsapi.h>

int qtoq_ioctl(
    int          descriptor,
    int          req_type,
    qos_req      *qos_data,
    unsigned int *qos_session,
    int          *qos_descriptor,
    )
```

Service Program Name: QSYS/QTOQSAPI

Default Public Authority: *EXCLUDE

Threadsafe: Yes

The **qtoq_ioctl()** API provides simplified Quality of Service functionality for connectionless sockets communications between RSVP aware applications on a client and server. This API can be used to initiate RSVP signaling, as well as to determine the status of the RSVP connection. The NO SIGNALLING option for loading RSVP rules also is supported.

Parameters

descriptor

(Input) Required

An opened socket descriptor that has been bound to the IP address and port that the application will use for connectionless communications.

req_type

(Input) Required

The type of QoS service being requested. The possible values are:

- | | |
|-----------------------------------|---|
| <i>REQ_SIGNAL_RET_EVENTS(1)</i> | Use normal RSVP signaling and return RSVP events to the calling program. |
| <i>REQ_SIGNAL_NORET_EVENTS(2)</i> | Use normal RSVP signaling without returning events to the calling program. |
| <i>REQ_NOSIGNAL(3)</i> | Load specified QoS policy if admission control allows it. |
| <i>REQ_GET_RSVP_DATA(4)</i> | Get the RSVP flowspec that has been returned as the result of an RSVP event. This request is valid only if a previous <i>REQ_SIGNAL_RET_EVENTS</i> request has been sent to the server. |

qos_data

(Input) Required

Pointer to a qos_req data structure that defines the type of service being requested and the source and

destination addresses of the request.

The qos_req data structure is defined below:

```
typedef struct
{
    int          service; ;    /* Values can be GUARANTEED_SERV (2)
                               or CONTROLLED_LOAD_SERV (5)
*/
    union
    {
        struct  CL_spec      /* Controlled-Load service      */
        {
            float    TB_Tspec_r; /* token bucket rate in bytes/sec */
            float    TB_Tspec_b; /* token bucket depth in bytes     */
            float    TB_Tspec_p; /* token bucket peak in bytes/sec  */
            unsigned long TB_Tspec_m; /* min policed unit in bytes     */
            unsigned long TB_Tspec_M; /* max packet size in bytes       */
        } CL_spec;
        struct Guar_spec    /* Guaranteed service          */
        {
            float    Guar_R;    /* guaranteed rate in bytes/sec   */
            unsigned long Guar_S; /* slack term in microseconds    */
        } Guar_spec;
    } spec_u;
} qos_spec_t;

typedef struct
{
    struct sockaddr    dest;    /* Destination address/port */
    int                d_length; /* Destination address length*/
    struct sockaddr    source;  /* Source address/port      */
    int                s_length; /* Source address length     */
    int                style;   /* Style of Reservation.    */
    qos_spec_t         Spec;    /* Flow info                */
    unsigned char      result;  /* API status               */
} qos_req; /* End of QoS request structure */
```

qos_session

(Output) Required

Pointer to an integer value where the unique QoS session ID can be stored. This ID is required for all future QoS API calls.

qos_descriptor

(Output) Optional

Pointer to an integer where the value of the descriptor that the application can wait on for RSVP events is stored. this value is set to NULL if it is not used.

Authorities

None.

Return Values

- 0* if successful.
- 1* if function failed. Errno indicates error reason.

Error Conditions

When this function call fails, the errno value is set to one of the following:

[EACCES]

Permission denied. This error code indicates one of the following:

- The process does not have the appropriate privileges to connect to the address pointed to by the *destination_address* parameter.
- The socket pointed to by *socket_descriptor* is using a connection-oriented transport service, and the *destination_address* parameter specifies a TCP/IP limited broadcast address (internet address of all ones).

[EADDRINUSE]

Address already in use. This error code indicates one of the following:

- The *socket_descriptor* parameter points to a connection-oriented socket that has been bound to a local address that contained no wildcard values, and the *destination_address* parameter specified an address that matched the bound address.
- The *socket_descriptor* parameter points to a socket that has been bound to a local address that contained no wildcard values, and the *destination_address* parameter (also containing no wildcard values) specified an address that would have resulted in a connection with an association that is not unique.

[EADDRNOTAVAIL]

Address not available. This error code indicates one of the following:

- The *socket_descriptor* parameter points to a socket with an address family of AF_INET and either a port was not available or a route to the address specified by the *destination_address* parameter could not be found.

[EAFNOSUPPORT]

The type of socket is not supported in this protocol family. The address family specified in the address structure pointed to by *destination_address* parameter cannot be used with the socket pointed to by the *socket_descriptor* parameter. This error also will be reported if the API is called with a socket type that is not AF_INET and SOCK_DGRAM or SOCK_STREAM.

[EALREADY]

Operation already in progress. A previous connect() function had already been issued for the socket pointed to by the *socket_descriptor* parameter, and has yet to be completed. This error code is returned only on sockets that use a connection-oriented transport service.

[EBADF]

Descriptor not valid.

[ECONNREFUSED]

The destination socket refused an attempted connect operation. This error occurs when there is no application that is bound to the address specified by the `destination_address` parameter.

[EFAULT]

Bad address. The system detected an address which was not valid while attempting to access the `destination_address` parameter.

[EHOSTUNREACH]

A route to the remote host is not available.

[EINPROGRESS]

Operation in progress. The `socket_descriptor` parameter points to a socket that is marked as non blocking and the connection could not be completed immediately. This error code is returned only on sockets that use a connection-oriented transport service.

[EINTR]

Interrupted function call.

[EINVAL]

Parameter not valid. This error code indicates one of the following:

- The `address_length` parameter specifies a length that is negative or not valid for the address family.
- The `AF_INET` socket is of type `SOCK_STREAM`, and a previous `connect()` has already completed unsuccessfully. Only one connection attempt is allowed on a connection-oriented socket.

[EIO]

Input/output error.

[EISCONN]

A connection has already been established.

This error code is returned only on sockets that use a connection-oriented transport service.

[ENETUNREACH]

Cannot reach the destination network. This error code indicates the following:

- For sockets that use the `AF_INET` address family, the address specified by the `destination_address` parameter requires the use of a router, and the socket option `SO_DONTROUTE` is currently set on.

[ENOBUFS]

There is not enough buffer space for the requested operation.

[ENOTDIR]

Not a directory.

[EOPNOTSUPP]

Operation not supported.

[ETIMEDOUT]

A remote host did not respond within the timeout period. This error code is returned when connection establishment times out. No connection is established. A possible cause may be that the partner application is bound to the address specified by the `destination_address` parameter, but the partner application has not yet issued a `listen()`.

[EUNKNOWN]

Unknown system state.

[EUNATCH]

The protocol required to support the specified address family is not available at this time.

[EPROTO]

An underlying protocol error has occurred.


Error Messages

| Message ID | Error Message Text |
|------------|--|
| CPE3418 E | Possible APAR condition or hardware failure. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |
| CPFA081 E | Unable to set return value or error code. |

Usage Notes

1. The application program can choose to be signaled when RSVP events occur or it can choose to allow the QoS server to handle the events. If the server handles the events, the application program will not be informed if the RSVP signaling failed or if the requested reservations was changed by the network.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#). 



API Introduced: V5R2

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

Open List of QoS Monitor Data (QgyOpenListQoSMonitorData) API

Required Parameter Group:

| | | | |
|---|-----------------------------|--------|-----------|
| 1 | Receiver variable | Output | Char(*) |
| 2 | Length of receiver variable | Input | Binary(4) |
| 3 | List information | Output | Char(80) |
| 4 | Number of records to return | Input | Binary(4) |
| 5 | Format name | Input | Char(8) |

Omissible Parameter Group:

| | | | |
|---|------------|-------|---------|
| 6 | Filter | Input | Char(*) |
| 7 | Error code | I/O | Char(*) |

Service Program: QSYS/QTOQMONAPI

Default Public Authority: *USE

Threadsafe: Yes

The **Open List of QoS Monitor Data (QgyOpenListQoSMonitorData)** API allows the user to gather information related to QoS services. Each entry is returned according to the particular FORMAT or type of filter selected. There are three types of data that can be retrieved: Instantaneous QoS Manager Data (stack), accumulated QoS Manager Data, or an aggregate of the accumulated QoS Manager Data.

Authorities and Locks

Special Authority

NONE

Required Parameter Group

Receiver Variable

OUTPUT; CHAR(*)

The receiver variable that receives the information requested.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable.

List information

OUTPUT; CHAR(80)

The variable used to return status information about the list of QoS monitor data that was opened. For a description of this parameter, see [Format of Open List Information](#).

Number of returned records

INPUT; BINARY(4)

The number of records in the list to put into the receiver variable after filtering has been performed.

Format name

INPUT; CHAR(8)

The format of the space information to be returned. The format names supported are:

- QOSM0100* IntServ controlled load and IntServ controlled load with DiffServ markings.
- » *QOSM0150* IntServ controlled load and IntServ controlled load with DiffServ markings. This format is used with 8-byte counters. «
- QOSM0200* IntServ guaranteed rate and IntServ guaranteed rate with DiffServ markings.
- » *QOSM0250* IntServ guaranteed rate and IntServ guaranteed rate with DiffServ markings. This format is used with 8-byte counters. «
- QOSM0300* DiffServ per hop behavior.
- » *QOSM0350* DiffServ per hop behavior. This format is used with 8-byte counters. «
- AGGR0100* IntServ controlled load and IntServ controlled load with DiffServ markings. Used for aggregated trace data only.
- » *AGGR0150* IntServ controlled load and IntServ controlled load with DiffServ markings. Used for aggregated trace data only. This format is used with 8-byte counters. «
- AGGR0200* IntServ guaranteed rate and IntServ guaranteed rate with DiffServ markings. Used for aggregated trace data only.
- » *AGGR0250* IntServ guaranteed rate and IntServ guaranteed rate with DiffServ markings. Used for aggregated trace data only. This format is used with 8-byte counters. «
- AGGR0300* DiffServ per hop behavior. Used for aggregated trace data only.
- » *AGGR0350* DiffServ per hop behavior. Used for aggregated trace data only. This format is used with 8-byte counters. «
- » *INBC0100* Connection information related to inbound IP policies. «
- » *INBC0200* Connection information related to URI inbound policies. «

Omissible Parameter Group

Filter

INPUT; CHAR(*)

The structure that defines which QoS filtered data is returned from the API.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see [Error Code Parameter](#) .

Filter Format Section

The following information is used for the filtering format. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|-------------------------|
| Dec | Hex | | |
| 0 | 0 | BINARY(4) | Length of filter |
| 4 | 4 | BINARY(4) | Filter flag |
| 8 | 8 | CHAR(14) | Start time |
| 22 | 16 | CHAR(14) | End time |
| 36 | 24 | BINARY(4) | Policy flag |
| 40 | 28 | BINARY(4) | System aggregation flag |
| 44 | 2C | CHAR(128) | Policy name |
| 172 | AC | CHAR(10) | Saved collected name |

Field Descriptions

End time. All data in the trace buffer between a given time interval. The format is YYYYMMDDHHMMSS, and HH should be represented with a 24-hour clock. If this parameter is set to a value, then the start time also must be set to some time less than the end time. This character string **must** be set to x'00' if not being used and other filtering parameters are needed.

Filter flag. Turns the filtering function on and off. The following values may be specified:

- 0 Turns off filtering option. If 0, the data is taken from the QoS manager (instantaneous stack data - QOSMxxxx format only).
- 1 Turns on filtering option. If 1, the data is taken from the trace buffer.

Length of filter. The length of the filtering structure.

Policy flag. Returns information for a specific policy for a given format. This option can be used in two ways. If it is used, Policy name must be set to some value.

- 1 Returns all entries for a specific policy within the user data for a given format. This option is used with the QOSMxxxx formats.
- 2 Return an aggregated list of a specific policy for given format. This option is used with AGGRxxxx formats.

The following values may be specified:

- 0 Turns off the policy option.
- 1 Turns on the policy option. System aggregation must be 0.

Policy name. Returns a list of entries associated with a given name. This option can be used only when the Policy flag is set to a value of 1. This character string must be set to x'00' if not being used and other filtering parameters are needed.

Start time. Returns all data in the trace buffer between a given time interval. The format is YYYYMMDDHHMMSS, and HH should be represented with a 24-hour clock. If this parameter is set to a value, the end time also must be set to some time greater than the start time. This character string must be set to x'00' if not being used and other filtering parameters are needed.

System aggregation flag. Returns a list of all aggregated policies within a given architecture. This filter option can be used with any of the AGGRxxxx formats only. The following values may be specified:

- 0 The system aggregation option is turned off. This value must be specified for QOSMxxxx formats.
- 1 The system aggregation option is turned on. Policy must be 0.

➤ **Saved collected name.** The collection data the user wishes to retrieve. If this value is blanks, or not supplied in the filter, then all data will be retrieved from the current data collection. ⚡

QOSM0100 Format

The QOSM0100 format includes the basic format of Integrated Services (IntServ) controlled load, and IntServ controlled load with Differentiated Services (DiffServ) markings. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|---|
| Dec | Hex | | |
| 0 | 0 | CHAR(128) | Policy name |
| 128 | 80 | CHAR(14) | Time stamp |
| 142 | 8E | CHAR(2) | Reserved - alignment |
| 144 | 90 | BINARY(4) | Protocol |
| 148 | 94 | CHAR(15) | Source IP address (start) - dotted decimal |
| 163 | A3 | CHAR(15) | Destination IP address (start) - dotted decimal |
| 178 | B2 | CHAR(15) | Source IP address (end) - dotted decimal |
| 193 | C1 | CHAR(15) | Destination IP address (end) - dotted decimal |
| 208 | D0 | BINARY(4) | Source port (start) |
| 212 | D4 | BINARY(4) | Destination port (start) |
| 216 | D8 | BINARY(4) | Source port (end) |
| 220 | DC | BINARY(4) | Destination port (end) |
| 224 | E0 | BINARY(4) | Token bucket rate - bytes per second |
| 228 | E4 | BINARY(4) | Token bucket depth - bytes |
| 232 | E8 | BINARY(4) | Peak data rate - bytes per second |
| 236 | EC | BINARY(4) | Minimum policed unit - bytes |

| | | | |
|-----|-----|-----------|--|
| 240 | F0 | BINARY(4) | Maximum packet size - bytes |
| 244 | F4 | BINARY(4) | Total connections serviced - connections |
| 248 | F8 | BINARY(4) | Total packets transmitted - packets |
| 252 | FC | BINARY(4) | Total bytes transmitted - bytes |
| 256 | 100 | BINARY(4) | Total in profile packets - packets |
| 260 | 104 | BINARY(4) | Total in profile bytes - bytes |

➤ QOSM0150 Format

The QOSM0150 format includes the basic format of Integrated Services (IntServ) controlled load, and IntServ controlled load with Differentiated Services (DiffServ) markings. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|--|
| Dec | Hex | | |
| 0 | 0 | CHAR(128) | Policy name |
| 128 | 80 | CHAR(14) | Time stamp |
| 142 | 8E | CHAR(2) | Reserved - alignment |
| 144 | 90 | BINARY(4) | Protocol |
| 148 | 94 | CHAR(15) | Source IP address - dotted decimal |
| 163 | A3 | CHAR(15) | Destination IP address - dotted decimal |
| 178 | B2 | CHAR(2) | Reserved - alignment |
| 180 | B4 | BINARY(4) | Source port (start) |
| 184 | B8 | BINARY(4) | Destination port (start) |
| 188 | BC | BINARY(4) | Token bucket rate - kbits per second |
| 192 | C0 | BINARY(4) | Token bucket depth - kbits |
| 196 | C4 | BINARY(4) | Peak data rate - kbits per second |
| 200 | C8 | BINARY(4) | Minimum policed unit - kbits |
| 204 | CC | BINARY(4) | Maximum packet size -kbits |
| 208 | D0 | BINARY(8) | Total packets transmitted long - packets |
| 216 | D8 | BINARY(8) | Total kbits transmitted long - kbits |
| 224 | E0 | BINARY(8) | Total in profile packets long - packets |
| 232 | E8 | BINARY(8) | Total in profile kbits long - kbits |
| 240 | F0 | BINARY(4) | Duration - seconds |
| 244 | F4 | BINARY(4) | Policy handle identifier |
| 248 | F8 | BINARY(4) | Offset to additional information |
| 252 | FC | BINARY(4) | Length of additional information➤ |

QOSM0200 Format

The QOSM0200 format includes the basic format of IntServ guaranteed rate and IntServ guaranteed rate with DiffServ markings (both). For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|------------------------------------|
| Dec | Hex | | |
| 0 | 0 | | Returns everything from QOSM0100 |
| 264 | 108 | BINARY(4) | Guaranteed rate - bytes per second |
| 268 | 10C | BINARY(4) | Slack term - second |

»QOSM0250 Format

The QOSM0250 format includes the basic format of IntServ guaranteed rate and IntServ guaranteed rate with DiffServ markings (both). For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|------------------------------------|
| Dec | Hex | | |
| | | | Offset from format QOSM0150 |
| 0 | 0 | BINARY(4) | Guaranteed rate - kbits per second |
| 4 | 4 | BINARY(4) | Slack term - second |

QOSM0300 Format

The QOSM0300 format includes the basic format of DiffServ per hop behavior. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|---|
| Dec | Hex | | |
| 0 | 0 | CHAR(128) | Policy name |
| 128 | 80 | CHAR(14) | Time stamp |
| 142 | 8E | CHAR(2) | Reserved - alignment |
| 144 | 90 | BINARY(4) | Priority |
| 148 | 94 | BINARY(4) | Protocol |
| 152 | 98 | CHAR(15) | Source IP address (start) - dotted decimal |
| 167 | A7 | CHAR(15) | Destination IP address (start) - dotted decimal |
| 182 | B6 | CHAR(15) | Source IP address (end) - dotted decimal |
| 197 | C5 | CHAR(15) | Destination IP address (end) - dotted decimal |
| 212 | D4 | BINARY(4) | Source port (start) |
| 216 | D8 | BINARY(4) | Destination port (start) |
| 220 | DC | BINARY(4) | Source port (end) |
| 224 | E0 | BINARY(4) | Destination port (end) |

| | | | |
|-----|-----|-----------|--|
| 228 | E4 | BINARY(4) | Token bucket rate - bytes per second |
| 232 | E8 | BINARY(4) | Token bucket depth - bytes |
| 236 | EC | BINARY(4) | Peak data rate - bytes per second |
| 240 | F0 | CHAR(1) | InDSCP |
| 241 | F1 | CHAR(1) | OutDSCP |
| 242 | F2 | CHAR(2) | Reserved - alignment |
| 244 | F4 | BINARY(4) | Total packets transmitted - packets |
| 248 | F8 | BINARY(4) | Total bytes transmitted - bytes |
| 252 | FC | BINARY(4) | Total in profile packets - packets |
| 256 | 100 | BINARY(4) | Total in profile bytes - bytes |
| 260 | 104 | BINARY(4) | Total active connections - connections |
| 264 | 108 | BINARY(4) | Traffic profile |

➤ QOSM0350 Format

The QOSM0350 format includes the basic format of DiffServ per hop behavior. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|---|
| Dec | Hex | | |
| 0 | 0 | CHAR(128) | Policy name |
| 128 | 80 | CHAR(14) | Time stamp |
| 142 | 8E | CHAR(2) | Reserved - alignment |
| 144 | 90 | BINARY(4) | Priority |
| 148 | 94 | BINARY(4) | Protocol |
| 152 | 98 | CHAR(15) | Source IP address (start) - dotted decimal |
| 167 | A7 | CHAR(15) | Destination IP address (start) - dotted decimal |
| 182 | B6 | CHAR(15) | Source IP address (end) - dotted decimal |
| 197 | C5 | CHAR(15) | Destination IP address (end) - dotted decimal |
| 212 | D4 | BINARY(4) | Source port (start) |
| 216 | D8 | BINARY(4) | Destination port (start) |
| 220 | DC | BINARY(4) | Source port (end) |
| 224 | E0 | BINARY(4) | Destination port (end) |
| 228 | E4 | BINARY(4) | Token bucket rate - kbits per second |
| 232 | E8 | BINARY(4) | Token bucket depth - kbits |
| 236 | EC | BINARY(4) | Peak data rate - kbits per second |
| 240 | F0 | CHAR(1) | InDSCP |
| 241 | F1 | CHAR(1) | OutDSCP |
| 242 | F2 | CHAR(2) | Reserved - alignment |
| 244 | F4 | BINARY(8) | Total packets transmitted long - packets |
| 252 | FC | BINARY(8) | Total kbits transmitted long - kbits |
| 260 | 104 | BINARY(8) | Total in profile packets long - packets |

| | | | |
|-----|-----|-----------|---|
| 268 | 10C | BINARY(8) | Total in profile kbits long - kbits |
| 276 | 114 | BINARY(8) | Total active connections long - connections |
| 284 | 11C | BINARY(4) | Traffic profile |
| 288 | 120 | BINARY(4) | Duration - seconds |
| 292 | 124 | BINARY(4) | Policy handle identifier |

AGGR0100 Format

The AGGR0100 format includes the basic format of IntServ controlled load and IntServ controlled load with DiffServ markings, and is used for aggregated trace data only. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|---------------------------------------|
| Dec | Hex | | |
| 0 | 0 | CHAR(128) | Policy name |
| 128 | 80 | BINARY(4) | Token bucket rate - bytes per second |
| 132 | 84 | BINARY(4) | Total connects serviced - connections |
| 136 | 88 | BINARY(4) | Total packets transmitted - packets |
| 140 | 8C | BINARY(4) | Total bytes transmitted - bytes |
| 144 | 90 | BINARY(4) | Total in profile packets - packets |
| 148 | 94 | BINARY(4) | Total in profile bytes - bytes |
| 152 | 98 | CHAR(14) | Start time |
| 166 | A6 | CHAR(14) | End time |

»AGGR0150 Format

The AGGR0150 format includes the basic format of IntServ controlled load and IntServ controlled load with DiffServ markings, and is used for aggregated trace data only. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|--|
| Dec | Hex | | |
| 0 | 0 | CHAR(128) | Policy name |
| 128 | 80 | BINARY(4) | Token bucket rate - kbits per second |
| 132 | 84 | BINARY(4) | Total connections serviced - connections |
| 136 | 88 | BINARY(8) | Total packets transmitted long - packets |
| 144 | 90 | BINARY(8) | Total kbits transmitted long - kbits |
| 152 | 98 | BINARY(8) | Total in profile packets long - packets |
| 160 | A0 | BINARY(8) | Total in profile kbits long - kbits |
| 168 | A8 | CHAR(14) | Start time |
| 182 | B6 | CHAR(14) | End time |

| | | | |
|-----|----|-----------|----------------------------------|
| 196 | C4 | BINARY(4) | Offset to additional information |
| 200 | C8 | BINARY(4) | Length of additional information |

AGGR0200 Format

The AGGR0200 format includes the basic format of IntServ guaranteed rate and IntServ guaranteed rate with DiffServ markings, and is used for aggregated trace data only. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|---|
| Dec | Hex | | |
| | | | Returns everything from AGGR0100 |
| 180 | B4 | BINARY(4) | Guaranteed rate - bytes per second |
| 184 | B8 | BINARY(4) | Actual calculated rate - bytes per second |

»AGGR0250 Format

The AGGR0250 format includes the basic format of IntServ guaranteed rate and IntServ guaranteed rate with DiffServ markings, and is used for aggregated trace data only. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|---|
| Dec | Hex | | |
| | | | Offset from format AGGR0150 |
| 0 | 0 | BINARY(4) | Guaranteed rate - kbits per second |
| 4 | 4 | BINARY(4) | Actual calculated rate - kbits per second |

AGGR0300 Format

The AGGR0300 Format includes the basic format of IntServ controlled load and IntServ controlled load with DiffServ markings, and is used for aggregated trace data only. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|--------------------------------------|
| Dec | Hex | | |
| 0 | 0 | CHAR(128) | Policy name |
| 128 | 80 | BINARY(4) | Token bucket rate - bytes per second |
| 132 | 84 | BINARY(4) | Token bucket depth - bytes |
| 136 | 88 | CHAR(1) | InDSCP |
| 137 | 89 | CHAR(1) | OutDSCP |

| | | | |
|-----|----|-----------|--|
| 138 | 8A | CHAR(2) | Reserved - alignment |
| 140 | 8C | BINARY(4) | Total active connections - connections |
| 144 | 90 | BINARY(4) | Total packets transmitted - packets |
| 148 | 94 | BINARY(4) | Total bytes transmitted - bytes |
| 152 | 98 | BINARY(4) | Total in profile packets - packets |
| 156 | 9C | BINARY(4) | Total in profile bytes - bytes |
| 160 | A0 | BINARY(4) | Traffic profile |
| 164 | A4 | CHAR(14) | Start time |
| 178 | B2 | CHAR(14) | End time |

➤AGGR0350 Format

The AGGR0350 Format includes the basic format of IntServ controlled load and IntServ controlled load with DiffServ markings, and is used for aggregated trace data only. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|---|
| Dec | Hex | | |
| 0 | 0 | CHAR(128) | Policy name |
| 128 | 80 | BINARY(4) | Token bucket rate - kbits per second |
| 132 | 84 | BINARY(4) | Token bucket depth - kbits |
| 136 | 88 | CHAR(1) | InDSCP |
| 137 | 89 | CHAR(1) | OutDSCP |
| 138 | 8A | CHAR(2) | Reserved - alignment |
| 140 | 8C | BINARY(8) | Total active connections long - connections |
| 148 | 94 | BINARY(8) | Total packets transmitted long - packets |
| 156 | 9C | BINARY(8) | Total kbits transmitted long - kbits |
| 164 | A4 | BINARY(8) | Total in profile packets long - packets |
| 172 | AC | BINARY(8) | Total in profile kbits long - kbits |
| 180 | B4 | BINARY(4) | Traffic profile |
| 184 | B8 | CHAR(14) | Start time |
| 198 | C6 | CHAR(14) | End time |

INBC0100 Format

The INBC0100 Format includes connection information related to inbound IP policies. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|-------------|
| Dec | Hex | | |
| 0 | 0 | CHAR(128) | Policy name |

| | | | |
|-----|-----|-----------|--|
| 128 | 80 | CHAR(14) | Time stamp |
| 142 | 8E | CHAR(2) | Reserved - alignment |
| 144 | 90 | BINARY(4) | Priority |
| 148 | 94 | CHAR(15) | Source IP address (start) - dotted decimal |
| 163 | A3 | CHAR(15) | Destination IP address (start) - dotted decimal |
| 178 | B2 | CHAR(15) | Source IP address (end) - dotted decimal |
| 193 | C1 | CHAR(15) | Destination IP address (end) - dotted decimal |
| 208 | D0 | BINARY(4) | Source port (start) |
| 212 | D4 | BINARY(4) | Destination port (start) |
| 216 | D8 | BINARY(4) | Source port (end) |
| 220 | DC | BINARY(4) | Destination port (end) |
| 224 | E0 | BINARY(4) | Average connection rate - connections per second |
| 228 | E4 | BINARY(4) | Connection burst - connections |
| 232 | E8 | BINARY(4) | Peak connection rate - connections per second |
| 236 | EC | BINARY(4) | Prioritized queue |
| 240 | F0 | BINARY(8) | Total connections transmitted - connections |
| 248 | F8 | BINARY(8) | Total in profile connections - connections |
| 256 | 100 | BINARY(4) | Duration - seconds |
| 260 | 104 | BINARY(4) | Policy handle identifier |

INBC0200 Format

The INBC0200 Format includes connection information related to URI inbound policies. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|-----------|---|
| Dec | Hex | | |
| 0 | 0 | CHAR(128) | Policy name |
| 128 | 80 | CHAR(14) | Time stamp |
| 142 | 8E | CHAR(2) | Reserved - alignment |
| 144 | 90 | BINARY(4) | Priority |
| 148 | 94 | CHAR(15) | Destination IP address (start) - dotted decimal |
| 163 | A3 | CHAR(15) | Destination IP address (end) - dotted decimal |
| 178 | B2 | CHAR(2) | Reserved - alignment |
| 180 | B4 | BINARY(4) | Destination port (start) |
| 184 | B8 | BINARY(4) | Average URI rate - URIs per second |
| 188 | BC | BINARY(4) | URI burst - number of URIs |
| 192 | C0 | BINARY(4) | Peak URI rate - URIs per second |
| 196 | C4 | BINARY(4) | Prioritized queue |
| 200 | C8 | BINARY(8) | Total URIs transmitted - number of URIs |
| 208 | D0 | BINARY(8) | Total in profile URIs - number of URIs |

| | | | |
|-----|----|-----------|--------------------------|
| 216 | D8 | BINARY(4) | Duration - seconds |
| 220 | DC | BINARY(4) | Policy handle identifier |
| 224 | E0 | CHAR(128) | URI name |

Field Descriptions

The field descriptions returned by this API for the various format types follows.

Actual calculated rate.. Actual calculated rate in bytes per second.

» **Average connection rate - connections per second.** The average number of new requests (connections) admitted per second.

Average URI rate - URIs per second. The average number of new URIs admitted per second.

Connection burst - number of connections. The maximum number of new requests (connections) accepted concurrently.

Destination IP address (end). The end of destination IP address range. IP address is in dotted decimal format.

Destination IP address (start). The start of the destination IP address range. IP address is in dotted decimal format. This value will be used if only one destination IP address is selected

Destination port (end). The end of the destination port range.

Destination port (start) The start of the destination port range. This value is used if only one port is selected

Duration. The Duration is the amount of time between the last query and the present query. This value is only set for Collected date.

End time. The ending time over which the aggregation was performed.

Guaranteed rate - bytes per second. The guaranteed rate in bytes per second.

InDSCP. The field used to select the per hop behavior (PHB) a packet will experience at each node.

Maximum packet size - bytes. The largest datagram that conforms to the traffic specifications.

Minimum policed unit - bytes. The smallest number of bytes that will be removed from the token bucket.

OutDSCP. The field used to select the per hop behavior (PHB) a packet will experience at each node.

» **Peak connection rate - connections per second.** The maximum allowable rate at which the source can inject connections into the network.

Peak data rate - bytes per second. The maximum rate at which the source and any reshaping point may inject burst of traffic into the network.

» **Peak URI rate - URI per second.** The maximum allowable rate at which the source can inject connections into the network.

Policy handle identifier. Is a unique handle for any given policy.❧

Policy name. The name of the policy with which the data is associated.

Priority. The priority assigned to each rule loaded in the QoS Manager.

❧**Prioritized queue -** The order the listen queue of the server processes incoming connections.❧

Protocol. The message protocol. Protocols may include:

6 TCP

17 UDP

255 RAW

Reserved - alignment. An ignored field.

Slack term - seconds. The difference between the desired delay and the delay obtained.

Source IP address (end). The end of the source IP address range. IP address is in dotted decimal format.

Source IP address (start). The start of the source IP address range. IP address is in dotted decimal format. This value is used if only one source IP address is selected

Source port (end). The end of the source port range.

Source port (start). The start of the source port range. This value is used if only one port is selected

Start time. The starting time over which the aggregation was performed.

Time stamp. The date and time the data was retrieved from the QoS Manager. The time is formatted with a 24-hour clock, and is in the format YYYYMMDDHHMMSS.

Token bucket depth - bytes. The number of tokens that can be stored in a given bucket.

Token bucket rate - bytes per second. The rate at which tokens can be sent into the network.

Total active connections. The total number of active connections.

❧**Total active connections long - connections** The total number of active connections. If this value is greater than 4,294,967,295 then the counter will wrap and start back at 1.❧

Total bytes transmitted - bytes. The total number of bytes transmitted. If this value is greater than 4,294,967,295 then the counter will wrap and start back at 1.

❧**Total bytes transmitted long - bytes** The total number of bytes transmitted.❧

Total connections serviced - number of connections. The total number of connections serviced.

❧**Total connections transmitted - number of connections** The total number of bytes transmitted.❧

Total in profile bytes - bytes. The total number of bytes transmitted in the profile. If this value is greater than 4,294,967,295 then the counter will wrap and start back at 1.

❧**Total in profile bytes long - bytes** The total number of bytes transmitted in profile.

Total in profile connections - connections. The total number of connection in the profile.❧

Total in profile packets - number of packets. The total number of in profile packets transmitted. If this value is greater than 4,294,967,295 then the counter will wrap and start back at 1.

➤**Total in profile packets long - number of packets** The total number of in profile packets transmitted.

Total in profile URIs - number of URIs. Total number of in profile URIs transmitted.⏪

Total packets transmitted - number of packets. The total number of packets transmitted. If this value is greater than 4,294,967,295 then the counter will wrap and start back at 1.

➤**Total packets transmitted long - number of packets** The total number of packets transmitted.

Total URIs transmitted - number of URIs. The total number of URIs transmitted.⏪

Traffic profile. The type of packet conditioning used on out-of-profile packets. The format may include:

- 1 Marking
- 2 Shaping
- 3 Dropping
- 4 Single marking

➤**URI burst - number of URIs.** The maximum number of new pages accepted concurrently.

URI name. A string of characters that represents the URI.⏪

Error Messages

| Message ID | Error Message Text |
|------------|--|
| TCP9215 E | QoS Monitor is active (not a valid state). |
| CPF0F03 E | Error in retrieving the user space that was created by the caller. |
| CPF24B4 E | Severe error while addressing parameter list. |
| CPF3C1E E | Required parameter 1 omitted. |
| CPF3C21 E | Format name 1 is not valid. |
| CPF3CF1 E | Error code parameter not valid. |
| CPF9802 E | Not authorized to object 2 in 3. |
| CPF9810 E | Library 1 not found. |
| CPF9820 E | Not authorized to use library 1. |
| CPF9872 E | Program or service program 1 in library 2 ended. Reason code 3. |

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

» Delete QoS Monitor Data (QtoqDeleteQoSMonitorData) API

Required Parameter Group:

| | | | |
|---|--|-------|-------------------|
| 1 | QoS collection name or names | Input | Array of Char(10) |
| 2 | Length of QoS collection name or names | Input | Binary(4) |
| 3 | Error code | I/O | Char(*) |

Service Program: QSYS/QTOQMONAPI

Default Public Authority: *USE

Threadsafe: Yes

The **Delete QoS Monitor Data (QtoqDeleteQoSMonitorData)** API allows the user to delete one or more sets of collected QoS monitor data.

Authorities and Locks

Special Authority

*IOSYSCFG

Required Parameter Group

QoS collection name or names

INPUT; CHAR(*)

The QoS collected name or names is an array of names the user wishes to delete.

Number of QoS collection name or names

INPUT; BINARY(4)

The length of the QoS collection names array. This value should be in multiples of 10.

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 |
|------------|--|
| CPF3C1E E | Required parameter &1 omitted. |
| CPF3CF1 E | Error code parameter not valid. |
| CPF9810 E | Library &1 not found. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |



API Introduced: V5R2

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

End QoS Monitor (QtoqEndQoSMonitor) API

Required Parameter Group:

1 Error Code I/O Char(*)

Service Program: QSYS/QTOQMONAPI

Default Public Authority: *USE

Threadsafe: Yes

The **End QoS Monitor (QtoqEndQoSMonitor) API** allows the user to stop gathering information related to QoS services.

Authorities and Locks

Special Authority

*IOSYSCFG

Required Parameter Group

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 |
|------------|--|
| TCP9216 E | QoS Monitor is not active (not a valid state). |
| CPF24B4 E | Severe error addressing parameter list. |
| CPF3C1E E | Required parameter &1 omitted. |
| CPF3CF1 E | Error code parameter not valid. |
| CPF3CF2 E | Error (s) occurred during running of &1 API. |

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

» List Saved QoS Monitor Data (QtoqListSavedQoSMonitorData) API

Required Parameter Group:

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

Service Program: QSYS/QTOQMONAPI

Default Public Authority: *USE

Threadsafe: Yes

The **List Saved QoS Monitor Data (QtoqListSavedQoSMonitorData)** API allows the user to return a list of all collected monitor data that was saved previously.

Authorities and Locks

Special Authority

NONE.

Required Parameter Group

Receiver Variable

OUTPUT; CHAR(*)

The receiver variable that receives the information requested.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable.

List information

OUTPUT; CHAR(80)

The variable used to return status information about the list of QoS monitor data that was opened. For a description of this parameter, see [Format of open list information](#).

Number of returned records

INPUT; BINARY(4)

The number of records in the list to put into the receiver variable.

Format name

INPUT; CHAR(8)

The format of the space information to be returned. The format name supported is:

QTOQ0100 Returns a list of names which are user spaces that contains QoS Monitor Data.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see [Error Code Parameter](#).

QTOQ0100 Format

The QTOQ0100 format includes the complete information for a saved QoS collected data object. For detailed descriptions of the fields in this table, see [Field Descriptions](#).

| Offset | | Type | Field |
|--------|-----|----------|---------------------------------|
| Dec | Hex | | |
| 0 | 0 | CHAR(10) | QoS collection name |
| 10 | A | CHAR(50) | QoS collection description text |

Field Descriptions

QoS collection description text. The QoS collection description text is a user-defined string of characters that are associated with each QoS collected monitor data.

QoS collection name. The name of each QoS collected monitor data found.

Error Messages

| Message ID | Error Message Text |
|------------|--|
| CPF3C1E E | Required parameter &1 omitted. |
| CPF3CF1 E | Error code parameter not valid. |
| CPF9810 E | Library &1 not found. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |



API Introduced: V5R2

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

» Save QoS Monitor Data (QtoqSaveQoSMonitorData) API

Required Parameter Group:

| | | | |
|---|---------------------|--------|----------|
| 1 | QoS collection name | Output | Char(10) |
| 2 | Description text | Input | Char(50) |
| 3 | Error code | I/O | Char(*) |

Service Program: QSYS/QTOQMONAPI

Default Public Authority: *USE

Threadsafe: Yes

The **Save QoS Monitor Data (QtoqSaveQoSMonitorData)** API allows the user to save a copy of the collected QoS monitor data for future use. The user is allowed to apply a description text field to the saved object of up to 50 characters. The actual name of the object is generated automatically by the API and returned to the user.

Authorities and Locks

Special Authority

*IOSYSCFG

Required Parameter Group

QoS collection name

OUTPUT; CHAR(10)

The QoS collection name automatically generated by the save command.

Description Text

INPUT; CHAR(50)

A user-friendly description of the collected data object that the user wishes to save.

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 |
|------------|--|
| CPF3C1E E | Required parameter &1 omitted. |
| CPF3CF1 E | Error code parameter not valid. |
| CPF9810 E | Library &1 not found. |
| CPF9872 E | Program or service program &1 in library &2 ended. Reason code &3. |



API Introduced: V5R2

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

Start QoS Monitor (QtoqStartQoSMonitor) API

Required Parameter Group:

| | | | |
|---|-------------|-------|-----------|
| 1 | Wrap | Input | Binary(4) |
| 2 | Buffer size | Input | Binary(4) |
| 3 | Granularity | Input | Binary(4) |
| 4 | Error Code | I/O | Char(*) |

Service Program: QSYS/QTOQMONAPI

Default Public Authority: *USE

Threadsafe: Yes

The **Start QoS Monitor (QtoqStartQoSMonitor) API** allows the user to gathering information related to QoS services.

Authorities and Locks

Special Authority

*IOSYSCFG

Required Parameter Group

Wrap

INPUT; BINARY(4)

Allows the user the option to continuously wrap the data buffer. The following values may be specified:

0 Do not wrap the buffer

1 Wrap the buffer

Buffer size

INPUT; BINARY(4)

The size of the buffer that will contain the user data.

Note: The buffer size is in kilobytes and can range from 16 to 16384.

Granularity

INPUT; BINARY(4)

The interval in seconds to update the trace information.

Note: The granularity is in seconds and can range from 5 to 86400.

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 |
|-------------------|--|
| TCP9215 E | QoS Monitor is active (not a valid state). |
| CPF24B4 E | Severe error addressing parameter list. |
| CPF3C1E E | Required parameter &1 omitted. |
| CPF3CF1 E | Error code parameter not valid. |
| CPF3CF2 E | Error (s) occurred during running of &1 API. |

API Introduced: V5R1

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

rapi_dispatch()--Dispatch the RAPI message-handling routine

Syntax

```
#include <rapi.h>
int rapi_dispatch(void)
```

Service program name: QSYS/QTOQRAPI

Default public authority: *USE

Threadsafe: Yes

The **rapi_dispatch()** API dispatches the RAPI message-handling routine defined in the `rapi_session()` call. The application should call this routine whenever a read event is signaled on a file descriptor returned by the **rapi_getfd()** API call. This routine may be called at any time, but generally it has no effect unless there is a pending event.

Calling this routine may result in one or more RAPI message-handling routines to the application from any of the Open API sessions known to this instance of the library.

Parameters

None.

Authorities

None.

Return Value

Returns 0 if successful.

RAPI error code if it fails.

Error Conditions

[*RAPI_ERR_NORSVP*] RSVP server was not detected. Make sure the RSVP server is running.


[RAPI_ERR_MEMFULL] Unable to allocate memory. Check the system memory status.

[RAPI_ERR_INVALID] Error detected in the RSVP specifications being handled by the RAPI message-handling routine.

Usage Notes

The **rapi_session()** API must be called to create a valid session before this API is called. This API typically is called to respond to an event received on the file descriptor returned by the **rapi_getfd()** API call.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#). 

Complete documentation of the RAPI APIs can be found at [The Open Group](#). 

API Introduced: V5R1

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

razi_fmt_adspec()--Format a RAPI Adspect

Syntax

```
#include <razi.h>
void razi_fmt_addspec(
    razi_adspec_t  *pAdspect,
    char           *pBuffer,
    int            size
)
```

Service program name: QSYS/QTOQRAPI

Default public authority: *USE

Threadsafe: Yes

The **razi_fmt_adspec()** API formats a RAPI Adspect into a string suitable for printing by converting the RAPI Adspect information that has been passed to the API into a string in the supplied buffer. The Adspect is a data element in the RSVP "path" message that carries a package of OPWA advertising information. This information contains data about the available end-to-end service available to the receivers of data and can be used to predict what service is available. The output string is truncated if the length of the string exceeds the buffer size.

Parameters

pAdspect

(Input) Required
A pointer to the Adspect to be formatted.

pBuffer

(Input/Output) Required
A pointer to the buffer to be used.

size

(Input) Required
The length of the supplied buffer.

Authorities

None.

Return Value

None.


Error Conditions

None.

Usage Notes

This API can be used to format the contents of the Adspec information that has been to the API into a string that can be displayed at the local output device.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#). 

Complete documentation of the RAPI APIs can be found at [The Open Group](#). 

API Introduced: V5R1

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

rapifmtfiltspec()--Format a RAPI Filter spec

Syntax

```
#include <rapifmtfiltspec.h>
void rapifmtfiltspec(
    rapi_filter_t *pFiltSpec,
    char *pBuffer,
    int size)
```

Service program name: QSYS/QTOQRAPI

Default public authority: *USE

Threadsafe: Yes

The **rapifmtfiltspec()** API formats a RAPI Filter spec into a string suitable for printing by converting the RAPI filter spec information that has been passed to the API into a string in the buffer that has been passed to the API. The filter spec defines the set of data packets that should receive the QoS defined in the flowspec. The output string is truncated if the length exceeds the buffer size.

Parameters

pFiltSpec

(Input) Required

A pointer to the filter spec structure to be formatted.

pBuffer

(Input/Output) Required

A pointer to the buffer to be used.

size

(Input) Required

The length of the supplied buffer.

Authorities

None.

Return Value

None.


Error Conditions

None.

Usage Notes

This API can be used to format the contents of the filter spec information that has been passed to the API into a string that can be displayed at the local output device.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#). 

Complete documentation of the RAPI APIs can be found at [The Open Group](#). 

API Introduced: V5R1

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

razi_fmt_flowspec()--Format a RAPI Flowspec

Syntax

```
#include <razi.h>
void razi_fmt_flowspec(
    razi_flowspec_t *pFlowspec,
    char *pBuffer,
    int size
```

Service program name: QSYS/QTOQRAPI

Default public authority: *USE

Threadsafe: Yes

The **razi_fmt_flowspec()** API formats a RAPI Flowspec into a string suitable for printing by converting the RAPI flowspec information that has been passed to the API into a character string in the buffer that was passed to the API. The flowspec defines the QoS that is to be provided to the data flow. The output string is truncated if the length of the string exceeds the buffer size.

Parameters

pFlowspec

(Input) Required
A pointer to the flowspec to be formatted.

pBuffer

(Input/Output) Required
A pointer to the buffer to be used.

size

(Input) Required
The length of the supplied buffer.

Authorities

None.

Return Value

None.


Error Conditions

None.

Usage Notes

This API can be used to format the contents of the Flowspec information that has been passed to the API into a string that can be displayed at the local output device.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#). 

Complete documentation of the RAPI APIs can be found at [The Open Group](#). 

API Introduced: V5R1

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

rapi_fmt_tspec()--Format a RAPI Tspec

Syntax

```
#include <rapi.h>

void rapi_fmt_tspec(
    rapi_tspec_t    *pTspec,
    char            *pBuffer,
    int              size
```

Service program name: QSYS/QTOQRAPI

Default public authority: *USE

Threadsafe: Yes

The **rapi_fmt_tspec()** formats a RAPI Tspec into a string suitable for printing by converting the RAPI Tspec information that has been passed to the API into a string in the buffer that has been passed to the API. The Tspec defines the traffic parameter set that defines a flow. The output string is truncated if the length of the string exceeds the buffer size.

Parameters

pTspec

(Input) Required
A pointer to the Tspec to be formatted.

pBuffer

(Input/Output) Required
A pointer to the buffer to be used.

size

(Input) Required
The length of the supplied buffer.

Authorities

None.

Return Value

None.


Error Conditions

None.

Usage Notes

This API can be used to format the contents of the Tspec information that has been passed to the API into a string that can be displayed at the local output device.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#). 

Complete documentation of the RAPI APIs can be found at [The Open Group](#). 

API Introduced: V5R1

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

rapi_getfd()--Get descriptor to wait on

Syntax

```
#include <rapi.h>
int rapi_getfd( rapi_sid_t SessID)
```

Service program name: QSYS/QTOQRAPI

Default public authority: *USE

Threadsafe: Yes

The **rapi_getfd()** API returns the file descriptor associated with a successful **rapi_session()** call. This descriptor is valid until **rapi_release()** has been called. When a read event is signaled on this file descriptor, the application should use **rapi_dispatch()** to call the RAPI message-handling routine to handle the event.

Parameters

SessID

(Input) Required

The session ID returned by a successful **rapi_session()** call.

Authorities

None.

Return Value

Returns a valid file descriptor if the SessID is valid.

Returns -1 if the SessID is not valid.


Error Conditions

None.

Usage Notes

The returned file descriptor can be used to wait on a `select()` or `poll()` call; it also can be used to wait on a `select()` call for a response from an API request. When the response is received, the `rapi_dispatch()` API can be used to call the RAPI message-handling routine defined in the `rapi_session()` call.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#). 

Complete documentation of the RAPI APIs can be found at [The Open Group](#). 

API Introduced: V5R1

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

rapi_release()--Release the currently active RAPI reservation

Syntax

```
#include <rapi.h>
int rapi_release( rapi_sid_t SessID )
```

Service program name: QSYS/QTOQRAPI

Default public authority: *USE

Threadsafe: Yes

The **rapi_release()** API releases the RAPI reservation that is active currently and closes the open sessions. This call is made implicitly if the application terminates without closing its RSVP sessions.

Parameters

SessID

(Input) Required

The session ID returned by a successful **rapi_session()** call.

Authorities

None.

Return Value

Returns 0 if successful.

Returns a RAPI error code if not successful.

Error Conditions


[*RAPI_ERR_BADSID*] The session ID passed to the API did not correspond to a valid RAPI session.

[*RAPI_ERR_NORSVP*] The RSVP server was not detected. Make sure the server has been started.

Usage Notes

1. The `rapi_session()` API must be called to establish a session ID to be used with the other RAPI APIs.
2. The RSVP server must be running before any of the RAPI APIs are called.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#). 

Complete documentation of the RAPI APIs can be found at [The Open Group](#). 

API Introduced: V5R1

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

rapi_reserve()--Make, modify, or delete a RAPI reservation

Syntax

```
#include <rapi.h>
int rapi_reserve(
    rapi_sid_t      SessID
    int             flags
    rapi_addr_t     *SessAddr
    rapi_styleid_t  style
    rapi_stylex_t   *style_ext
    rapi_policy_t   *RcvPol
    int             numFilt
    rapi_filter_t   *FspecLst
    int             numFlow
    rapi_flowspec_t *Flowlst
```

Service program name: QSYS/QTOQRAPI

Default public authority: *USE

Threadsafe: Yes

The RSVP receiver uses the the **rapi_reserve()** API to make, modify, or delete an RSVP reservation in the network. This call causes an RSVP RESERVE message to be sent to the sender through the network. This API should be called after a PATH message has been received from the sender.

Parameters

SessID

(Input) Required

Session ID returned by a successful **rapi_session()** call.

flags

(Input) Required

Set to 0 if not used.

RAPI_REQ_CONFIRM 32

- Requests confirmation of the reservation by means of a confirmation RAPI message-handling routine (type RAPI_RESV_CONFIRM).

SessAddr

(Input) Required

A pointer to a rapi_addr_t structure that defines the interface address to receive data for multicast flows. If omitted or the host address is INADDR_ANY, the default interface is assumed. It is set to

0 if not used.

style

(Input) Required
A reservation style ID (see table below).

style_ext

(Input) Optional
A pointer to a style-dependent extension to the parameter list if there is one. Otherwise, it is NULL.

RcvPol

(Input) Optional
A pointer to a policy data structure. It is set to NULL if not used.

NumFilt

(Input) Required
The number of filter specs. If the NumFilt parameter is 0, the FspecLst parameter is ignored.

FspecLst

(Input) Optional
A pointer to an area containing a sequential vector of RAPI filter spec objects. It is set to NULL if not used.

numFlow

(Input) Required
The number of flow specs. If numFlow is zero, the call removes the current reservations for the specified session and FSpecLst. The FlowLst parameter will be ignored.

FlowLst

(Input) Optional
A pointer to an area containing a sequential vector of RAPI flowspec objects. The number of objects is specified in the numFlow parameter. If the numFlow parameter is 0, this input is ignored and should be set to NULL.

RAPI Styles

| Style Type | Style ID | Description |
|---------------------|----------------------|---|
| Wildcard Filter(WF) | RAPI_RSTYLE_WILDCARD | The Flowspec_list parameter may be empty (to delete the reservation) or else point to a single flowspec. The FilterSpec_list parameter may be empty or it may point to a single filter spec containing appropriate wildcard(s). |
| Fixed Filter(FF) | RAPI_RSTYLE_FIXED | FilterSpecNo must equal FlowspecNo. Entries Flowspec_list and FilterSpec_list parameters will correspond in pairs. |

| | | |
|---------------------|------------|---|
| Shared Explicit(SE) | _RSTYLE_SE | The Flowspec_list parameter should point to a single flowspec. The FilterSpec_list parameter may point to a list of any length. |
|---------------------|------------|---|

Authorities

None.

Return Value

Returns 0 if successful.

RAPI error code if it fails.

Error Conditions

[*RAPI_ERR_INVALID*] One or more of the parameters that was passed to the API was not valid.


[*RAPI_ERR_BADSID*] The session ID that was passed to the API did not correspond to an active RAPI session.

[*RAPI_ERR_NORSVP*] The RSVP server was not detected. Make sure the RSVP server is running.

Usage Notes

If this call is successful, the application RAPI message-handling routine of type RAPI_RESV_ERROR or RAPI_RESV_CONFIRM may be generated. A rejection of the reservation request or other failure is reported by an RAPI message-handling routine of type API_RESV_ERROR. An error code of RSPV_Err_NO_PATH indicates that the RSVP state from one or more of the senders specified in filter_list has not yet propagated all the way to the receiver; it may also indicate that one or more of the specified senders has closed its API session and that its RSVP state has been deleted from the routers.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#). 

Complete documentation of the RAPI APIs can be found at [The Open Group](#). 

API Introduced: V5R1

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

razi_sender()--Identify a RAPI sender

Syntax

```
#include <razi.h>

int razi_sender (
    razi_sid_t      SessID,
    int             flags,
    razi_addr_t     *LocalAddr,
    razi_filter_t   *Filter,
    razi_tspec_t    *SndTspec,
    razi_adspec_t   *SndAdspec,
    razi_policy_t   *SndPol,
    int             Ttl
```

Service program name: QSYS/QTOQRAPI

Default public authority: *USE

Threadsafe: Yes

The **razi_sender()** API identifies an RSVP sender to potential receivers of the data. This API causes an RSVP path message to be sent to the receiver defined by the SessID value obtained by a **razi_session** call.

Parameters

SessID

(Input) Required

The session ID returned by a successful **razi_session()** call. A session ID that is not valid will cause the API to fail.

flags

(Input) Required

Either a zero or the following flags may be used.

- | | | |
|----|-----------------|----------------------------|
| 2 | TC_QOS_POLICE | Turn traffic policing on. |
| 4 | TC_QOS_NOPOLICE | Turn traffic policing off. |
| 8 | TC_QOS_SHAPE | Turn traffic shaping on. |
| 16 | TC_QOS_NOSHAPE | Turn traffic shaping off. |

LocalAddr

(Input) Optional

A pointer to a `rapi_addr_t` structure defining the IP source address and, if needed, the source port or flow label from which data will be sent. It is set to NULL if not used. The format of a `rapi_addr_t` is implementation-dependent.

Filter

(Input) Optional

A pointer to a RAPI filter spec structure defining the format of the data packets to be sent. It is set to NULL if not used. If this parameter is NULL, a sender template is created internally from the *Dest* and *LocalAddr* parameters. The *Dest* parameter was provided as part of the `rapi_session()` call. If the *Filter* parameter is present, the *LocalAddr* parameter is ignored.

If the session is using IPSEC, this parameter is required.

SndTspec

(Input) Required

A pointer to a Tspec that defines the traffic this sender will create.

SndAdspec

(Input) Optional

A pointer to a RAPI Adspec structure. It is set to NULL if not used.

SndPol

(Input) Optional

A pointer to a sender policy data structure. It is set to NULL if not used.

Ttl

(Input) Required

The IP TTL (Time-to-Live) value for sending multicast data. It allows RSVP to send its control messages with the same TTL scope as the data packets. It is set to 0 if not used.

Authorities

None.

Return Value

Returns 0 if successful.

A RAPI error code is returned if it fails.


Error Conditions

- [RAPI_ERR_INVALID]* A parameter that is not valid was passed to the API.
- [RAPI_ERR_BADSID]* The session ID passed to the API was valid.
- [RAPI_ERR_NOTSPEC]* No sender Tspec was defined for the API call.
- [RAPI_ERR_NORSVP]* The RSVP server did not respond to the API request. Make sure the RSVP server is running.

Usage Notes

1. The **rapi_session()** API must be called to establish a session ID to be used with the other RAPI API's.
2. The RSVP server must be running before any of the RAPI APIs are called.
3. The formats of the parameter structures are defined in the **<rapi.h>** header file.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#). 

Complete documentation of the RAPI APIs can be found at [The Open Group](#). 

API Introduced: V5R1

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

rapi_session()--Create a RAPI session

Syntax

```
#include <rapi.h>
rapi_sid_t rapi_session(
    rapi_addr_t    *Dest,
    int            Protid,
    int            flags,
    rapi_event_rtn_t Event_rtn,
    void           *Event_arg,
    int            *errno)
```

Service program name: QSYS/QTOQRAPI

Default public authority: *USE

Threadsafe: Yes

The **rapi_session()** API returns an API session ID that is unique to this request. This ID is used in calling the other RAPI APIs to identify which RSVP session is being requested.

Parameters

Dest

(Input) Required

A pointer to a `rapi_addr_t` structure defining the destination IP address and a port number that is the target of the data. The *dest* and *protid* parameters are used to identify an RSVP session. If the *protid* specifies UDP or TCP transport, the port value identifies the appropriate transport port number. The format of the `rapi_addr_t` structure is implementation-dependent.

Protid

(Input) Required

The IP protocol ID for the session. This value can be either 17(UDP) or 6(TCP). If it is zero, then 17(UDP) is assumed.

flags

(Input) Required

The flags value is set as follows:

- 1 RAPI_USE_INTSERV Currently, the only flag supported. If this flag is set, IntServ formats are used in the RAPI message-handling routines. If this flag is not set, the simplified format is used.

Event_rtn

(Input) Required

A pointer to a RAPI message-handling routine that is called to communicate RSVP errors and state change events to the calling application. The RAPI message-handling routine is called when the **rapi_dispatch()** API is called as the result of events. This pointer is used with select() or poll(). This routine must be supplied by the application calling the API.

Event_arg

(Input/Output) Optional

An argument data that is passed to the RAPI message-handling routine function when it is called. It is set to NULL if not used.

errnop

(Input/Output) Required

A pointer to an integer in which a RAPI error code will be returned.

Authorities

None

Return Value

Successful completion of this function returns a nonzero session handle that can be used in further RAPI calls for this session. If the call fails, it returns zero (RAPI_NULL_SID) and stores a RAPI error code in the integer errnop parameter.

Error Conditions

[RAPI_ERR_NORSVP] The RSVP server did not respond to the API request. Make sure the RSVP server is running.

[RAPI_ERR_UNSUPPORTED] The flags parameter was set to an unsupported value.

[RAPI_ERR_SYSCALL] There was a problem calling a system function. Check the system errno for further information.


[RAPI_ERR_INVALID] A parameter that is not valid was used in the API call.

[RAPI_ERR_MAXSESS] The maximum number of available RAPI sessions has been exceeded.

Usage Notes

1. The `rapi_session()` API must be called to establish a session ID to be used with the other RAPI APIs.
2. The RSVP server must be running before any of the RAPI APIs are called.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#). 

Complete documentation of the RAPI APIs can be found at [The Open Group](#). 

API Introduced: V5R1

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

rapi_version()--Retrieve the current RAPI version

Syntax

```
#include <rapi.h>
int rapi_version(void)
```

Service program name: QSYS/QTOQRAPI

Default public authority: *USE

Threadsafe: Yes

The **rapi_version()** API returns the RAPI version currently being used by the RSVP agent.

Parameters

None.

Authorities

None.

Return Value

An integer representing the version number of the RAPI interface. The value defines a major and minor number that is encoded as "100*major + minor".


Error Conditions

None.

Usage Note

None.

Related Information

For a description of the RSVP protocol, see RFC 2205 on the RFC Pages for [The Internet Engineering Task Force](#). 

Complete documentation of the RAPI APIs can be found at [The Open Group](#). 

API Introduced: V5R1

[Top](#) | [UNIX-Type APIs](#) | [APIs by category](#)

Header Files for UNIX-Type Functions

Programs using the UNIX-type functions must include one or more header files that contain information needed by the functions, such as:

- Macro definitions
- Data type definitions
- Structure definitions
- Function prototypes

The header files are provided in the QSYSINC library, which is optionally installable. Make sure QSYSINC is on your system before compiling programs that use these header files. For information on installing the QSYSINC library, see [Data structures and the QSYSINC Library](#).

The table below shows the file and member name in the QSYSINC library for each header file used by the UNIX-type APIs in this publication.

| Name of Header File | Name of File in QSYSINC | Name of Member |
|---------------------|-------------------------|----------------|
| arpa/inet.h | ARPA | INET |
| arpa/nameser.h | ARPA | NAMESER |
| bse.h | H | BSE |
| bsedos.h | H | BSEDOS |
| bseerr.h | H | BSEERR |
| dirent.h | H | DIRENT |
| errno.h | H | ERRNO |
| fcntl.h | H | FCNTL |
| grp.h | H | GRP |
| »inttypes.h | H | INTTYPES« |
| limits.h | H | LIMITS |
| »mman.h | H | MMAN« |
| netdb.h | H | NETDB |
| »netinet/icmp6.h | NETINET | ICMP6« |
| net/if.h | NET | IF |
| netinet/in.h | NETINET | IN |
| netinet/ip_icmp.h | NETINET | IP_ICMP |
| netinet/ip.h | NETINET | IP |
| »netinet/ip6.h | NETINET | IP6« |
| netinet/tcp.h | NETINET | TCP |
| netinet/udp.h | NETINET | UDP |
| netns/idp.h | NETNS | IDP |
| netns/ipx.h | NETNS | IPX |
| netns/ns.h | NETNS | NS |
| netns/sp.h | NETNS | SP |
| net/route.h | NET | ROUTE |
| nettel/tel.h | NETTEL | TEL |

| | | |
|-----------------|-----|-----------|
| os2.h | H | OS2 |
| os2def.h | H | OS2DEF |
| pwd.h | H | PWD |
| Qlg.h | H | QLG |
| qp0lflop.h | H | QP0LFLOP |
| »qp0ljrnl.h | H | QP0LJRNL« |
| »qp0lrord.h | H | QP0LRORD« |
| Qp0lstdi.h | H | QP0LSTDI |
| qp0wpid.h | H | QP0WPID |
| qp0zdipc.h | H | QP0ZDIPC |
| qp0zipc.h | H | QP0ZIPC |
| qp0zolip.h | H | QP0ZOLIP |
| qp0zolsm.h | H | QP0ZOLSM |
| qp0zripc.h | H | QP0ZRIPC |
| qp0ztrc.h | H | QP0ZTRC |
| qp0ztrml.h | H | QP0ZTRML |
| qp0z1170.h | H | QP0Z1170 |
| »qsoasync.h | H | QSOASYNC« |
| qtnxaapi.h | H | QTNXAAPI |
| qtnxadtp.h | H | QTNXADTP |
| qtomeapi.h | H | QTOMEAPI |
| qtossapi.h | H | QTOSSAPI |
| resolv.h | H | RESOLVE |
| semaphore.h | H | SEMAPHORE |
| signal.h | H | SIGNAL |
| spawn.h | H | SPAWN |
| ssl.h | H | SSL |
| sys/errno.h | H | ERRNO |
| sys/ioctl.h | SYS | IOCTL |
| sys/ipc.h | SYS | IPC |
| sys/layout.h | H | LAYOUT |
| sys/limits.h | H | LIMITS |
| sys/msg.h | SYS | MSG |
| sys/param.h | SYS | PARAM |
| »sys/resource.h | SYS | RESOURCE« |
| sys/sem.h | SYS | SEM |
| sys/setjmp.h | SYS | SETJMP |
| sys/shm.h | SYS | SHM |
| sys/signal.h | SYS | SIGNAL |
| sys/socket.h | SYS | SOCKET |
| sys/stat.h | SYS | STAT |
| sys/statvfs.h | SYS | STATVFS |

| | | |
|----------------------------|-----|--------------------------|
| sys/time.h | SYS | TIME |
| sys/types.h | SYS | TYPES |
| sys/uio.h | SYS | UIO |
| sys/un.h | SYS | UN |
| sys/wait.h | SYS | WAIT |
| » ulimit.h | H | ULIMIT « |
| unistd.h | H | UNISTD |
| utime.h | H | UTIME |

You can display a header file in QSYSINC by using one of the following methods:

- Using your editor. For example, to display the **unistd.h** header file using the Source Entry Utility editor, enter the following command:

```
STRSEU SRCFILE(QSYSINC/H) SRCMBR(UNISTD) OPTION(5)
```

- Using the Display Physical File Member command. For example, to display the **sys/stat.h** header file, enter the following command:

```
DSPPFM FILE(QSYSINC/SYS) MBR(STAT)
```

You can print a header file in QSYSINC by using one of the following methods:

- Using your editor. For example, to print the **unistd.h** header file using the Source Entry Utility editor, enter the following command:

```
STRSEU SRCFILE(QSYSINC/H) SRCMBR(UNISTD) OPTION(6)
```

- Using the Copy File command. For example, to print the **sys/stat.h** header file, enter the following command:

```
CPYF FROMFILE(QSYSINC/SYS) TOFILE(*PRINT) FROMMBR(STAT)
```

Symbolic links to these header files are also provided in directory /QIBM/include.

Errno Values for UNIX-Type Functions

Programs using the UNIX-type functions may receive error information as *errno* values. The possible values returned are listed here in ascending *errno* value sequence.

| Name | Value | Text |
|-----------|-------|--|
| EDOM | 3001 | A domain error occurred in a math function. |
| ERANGE | 3002 | A range error occurred. |
| ETRUNC | 3003 | Data was truncated on an input, output, or update operation. |
| ENOTOPEN | 3004 | File is not open. |
| ENOTREAD | 3005 | File is not opened for read operations. |
| EIO | 3006 | Input/output error. |
| ENODEV | 3007 | No such device. |
| ERECIO | 3008 | Cannot get single character for files opened for record I/O. |
| ENOTWRITE | 3009 | File is not opened for write operations. |
| ESTDIN | 3010 | The stdin stream cannot be opened. |
| ESTDOUT | 3011 | The stdout stream cannot be opened. |
| ESTDERR | 3012 | The stderr stream cannot be opened. |
| EBADSEEK | 3013 | The positioning parameter in fseek is not correct. |
| EBADNAME | 3014 | The object name specified is not correct. |
| EBADMODE | 3015 | The type variable specified on the open function is not correct. |
| EBADPOS | 3017 | The position specifier is not correct. |
| ENOPOS | 3018 | There is no record at the specified position. |
| ENUMMBRS | 3019 | Attempted to use ftell on multiple members. |
| ENUMRECS | 3020 | The current record position is too long for ftell. |
| EINVAL | 3021 | The value specified for the argument is not correct. |
| EBADFUNC | 3022 | Function parameter in the signal function is not set. |
| ENOENT | 3025 | No such path or directory. |
| ENOREC | 3026 | Record is not found. |
| EPERM | 3027 | The operation is not permitted. |
| EBADDATA | 3028 | Message data is not valid. |
| EBUSY | 3029 | Resource busy. |
| EBADOPT | 3040 | Option specified is not valid. |
| ENOTUPD | 3041 | File is not opened for update operations. |
| ENOTDLT | 3042 | File is not opened for delete operations. |

| | | |
|---------------|------|--|
| EPAD | 3043 | The number of characters written is shorter than the expected record length. |
| EBADKEYLN | 3044 | A length that was not valid was specified for the key. |
| EPUTANDGET | 3080 | A read operation should not immediately follow a write operation. |
| EGETANDPUT | 3081 | A write operation should not immediately follow a read operation. |
| EIOERROR | 3101 | A nonrecoverable I/O error occurred. |
| EIORECERR | 3102 | A recoverable I/O error occurred. |
| EACCES | 3401 | Permission denied. |
| ENOTDIR | 3403 | Not a directory. |
| ENOSPC | 3404 | No space is available. |
| EXDEV | 3405 | Improper link. |
| EAGAIN | 3406 | Operation would have caused the process to be suspended. |
| EWOULDBLOCK | 3406 | Operation would have caused the process to be suspended. |
| EINTR | 3407 | Interrupted function call. |
| EFAULT | 3408 | The address used for an argument was not correct. |
| ETIME | 3409 | Operation timed out. |
| ENXIO | 3415 | No such device or address. |
| EAPAR | 3418 | Possible APAR condition or hardware failure. |
| ERECURSE | 3419 | Recursive attempt rejected. |
| EADDRINUSE | 3420 | Address already in use. |
| EADDRNOTAVAIL | 3421 | Address is not available. |
| EAFNOSUPPORT | 3422 | The type of socket is not supported in this protocol family. |
| EALREADY | 3423 | Operation is already in progress. |
| ECONNABORTED | 3424 | Connection ended abnormally. |
| ECONNREFUSED | 3425 | A remote host refused an attempted connect operation. |
| ECONNRESET | 3426 | A connection with a remote socket was reset by that socket. |
| EDESTADDRREQ | 3427 | Operation requires destination address. |
| EHOSTDOWN | 3428 | A remote host is not available. |
| EHOSTUNREACH | 3429 | A route to the remote host is not available. |
| EINPROGRESS | 3430 | Operation in progress. |
| EISCONN | 3431 | A connection has already been established. |
| EMSGSIZE | 3432 | Message size is out of range. |
| ENETDOWN | 3433 | The network currently is not available. |
| ENETRESET | 3434 | A socket is connected to a host that is no longer available. |

| | | |
|-----------------|------|--|
| ENETUNREACH | 3435 | Cannot reach the destination network. |
| ENOBUFS | 3436 | There is not enough buffer space for the requested operation. |
| ENOPROTOPT | 3437 | The protocol does not support the specified option. |
| ENOTCONN | 3438 | Requested operation requires a connection. |
| ENOTSOCK | 3439 | The specified descriptor does not reference a socket. |
| ENOTSUP | 3440 | Operation is not supported. |
| EOPNOTSUPP | 3440 | Operation is not supported. |
| EPFNOSUPPORT | 3441 | The socket protocol family is not supported. |
| EPROTONOSUPPORT | 3442 | No protocol of the specified type and domain exists. |
| EPROTOTYPE | 3443 | The socket type or protocols are not compatible. |
| ERCVDERR | 3444 | An error indication was sent by the peer program. |
| ESHUTDOWN | 3445 | Cannot send data after a shutdown. |
| ESOCKTNOSUPPORT | 3446 | The specified socket type is not supported. |
| ETIMEDOUT | 3447 | A remote host did not respond within the timeout period. |
| EUNATCH | 3448 | The protocol required to support the specified address family is not available at this time. |
| EBADF | 3450 | Descriptor is not valid. |
| EMFILE | 3452 | Too many open files for this process. |
| ENFILE | 3453 | Too many open files in the system. |
| EPIPE | 3455 | Broken pipe. |
| ECANCEL | 3456 | Operation cancelled. |
| EEXIST | 3457 | File exists. |
| EDEADLK | 3459 | Resource deadlock avoided. |
| ENOMEM | 3460 | Storage allocation request failed. |
| EOWNERTERM | 3462 | The synchronization object no longer exists because the owner is no longer running. |
| EDESTROYED | 3463 | The synchronization object was destroyed, or the object no longer exists. |
| ETERM | 3464 | Operation was terminated. |
| ENOENT1 | 3465 | No such file or directory. |
| ENOEQFLOG | 3466 | Object is already linked to a dead directory. |
| EEMPTYDIR | 3467 | Directory is empty. |
| EMLINK | 3468 | Maximum link count for a file was exceeded. |

| | | |
|--------------|------|--|
| ESPIPE | 3469 | Seek request is not supported for object. |
| ENOSYS | 3470 | Function not implemented. |
| EISDIR | 3471 | Specified target is a directory. |
| EROFS | 3472 | Read-only file system. |
| EUNKNOWN | 3474 | Unknown system state. |
| EITERBAD | 3475 | Iterator is not valid. |
| EITERSTE | 3476 | Iterator is in wrong state for operation. |
| EHRICLSBAD | 3477 | HRI class is not valid. |
| EHRICLBAD | 3478 | HRI subclass is not valid. |
| EHRITYPBAD | 3479 | HRI type is not valid. |
| ENOTAPPL | 3480 | Data requested is not applicable. |
| EHRIREQTYP | 3481 | HRI request type is not valid. |
| EHRINAMEBAD | 3482 | HRI resource name is not valid. |
| EDAMAGE | 3484 | A damaged object was encountered. |
| ELOOP | 3485 | A loop exists in the symbolic links. |
| ENAMETOOLONG | 3486 | A path name is too long. |
| ENOLCK | 3487 | No locks are available. |
| ENOTEMPTY | 3488 | Directory is not empty. |
| ENOSYSRSC | 3489 | System resources are not available. |
| ECONVERT | 3490 | Conversion error. |
| E2BIG | 3491 | Argument list is too long. |
| EILSEQ | 3492 | Conversion stopped due to input character that does not belong to the input codeset. |
| ETYPE | 3493 | Object type mismatch. |
| EBADDIR | 3494 | Attempted to reference a directory that was not found or was destroyed. |
| EBADOBJ | 3495 | Attempted to reference an object that was not found, was destroyed, or was damaged. |
| EIDXINVAL | 3496 | Data space index used as a directory is not valid. |
| ESOFTDAMAGE | 3497 | Object has soft damage. |
| ENOTENROLL | 3498 | User is not enrolled in system distribution directory. |
| EOffline | 3499 | Object is suspended. |
| EROOBJ | 3500 | Object is a read-only object. |
| EEAHDDSI | 3501 | Hard damage on extended attribute data space index. |
| EEASDDSI | 3502 | Soft damage on extended attribute data space index. |
| EEAHDDS | 3503 | Hard damage on extended attribute data space. |
| EEASDDS | 3504 | Soft damage on extended attribute data space. |
| EEADUPRC | 3505 | Duplicate extended attribute record. |

| | | |
|----------------|------|--|
| ELOCKED | 3506 | Area being read from or written to is locked. |
| EFBIG | 3507 | Object too large. |
| EIDRM | 3509 | The semaphore, shared memory, or message queue identifier is removed from the system. |
| ENOMSG | 3510 | The queue does not contain a message of the desired type and (msgflg logically ANDed with IPC_NOWAIT). |
| EFILECVT | 3511 | File ID conversion of a directory failed. |
| EBADFID | 3512 | A file ID could not be assigned when linking an object to a directory. |
| ESTALE | 3513 | File handle was rejected by server. |
| ESRCH | 3515 | No such process. |
| ENOTSIGINIT | 3516 | Process is not enabled for signals. |
| ECHILD | 3517 | No child process. |
| EBADH | 3520 | Handle is not valid. |
| ETOOMANYREFS | 3523 | The operation would have exceeded the maximum number of references allowed for a descriptor. |
| ENOTSAFE | 3524 | Function is not allowed. |
| E_OVERFLOW | 3525 | Object is too large to process. |
| EJRNDDAMAGE | 3526 | Journal is damaged. |
| EJRNINACTIVE | 3527 | Journal is inactive. |
| EJRNRCVSPC | 3528 | Journal space or system storage error. |
| EJRNRMNT | 3529 | Journal is remote. |
| ENEWJRNRCV | 3530 | New journal receiver is needed. |
| ENEWJRN | 3531 | New journal is needed. |
| EJOURNALED | 3532 | Object already journaled. |
| EJRNENTTOOLONG | 3533 | Entry is too large to send. |
| EDATALINK | 3534 | Object is a datalink object. |
| ENOTAVAIL | 3535 | IASP is not available. |
| ENOTTY | 3536 | I/O control operation is not appropriate. |
| EFBIG2 | 3540 | Attempt to write or truncate file past its sort file size limit. |
| ETXTBSY | 3543 | Text file busy. |
| EASPGRPNOTSET | 3544 | ASP group not set for thread. |
| ERESTART | 3545 | A system call was interrupted and may be restarted. |