Before using this information and the product it supports, be sure to read the information in "Notices," on page 49.
Contents

Resource Reservation Setup Protocol

APIs ........................................... 1

APIs ........................................... 2
Open List of QoS Monitor Data
(QtoqOpenListQoSMonitorData) API ... 2
Authority and Locks ..................... 3
Required Parameter Group ............. 3
Omissible Parameter Group .......... 3
Filter Format Section .................. 4
Field Descriptions ...................... 4
QOSM0100 Format ..................... 5
QOSM0150 Format ..................... 6
QOSM0200 Format ..................... 6
QOSM0250 Format ..................... 7
QOSM0300 Format ..................... 7
QOSM0350 Format ..................... 8
AGGR0100 Format ...................... 8
AGGR0150 Format ...................... 9
AGGR0200 Format ...................... 9
AGGR0250 Format ..................... 10
AGGR0300 Format ..................... 10
AGGR0350 Format ..................... 10
INBC0100 Format ..................... 11
INBC0200 Format ..................... 12
Field Descriptions .................... 12
Error Messages ....................... 15
Delete QoS Monitor Data
(QtoqDeleteQoSMonitorData) API ..... 15
Authority and Locks .................. 15
Required Parameter Group .......... 15
Error Messages ....................... 16
End QoS Monitor (QtoqEndQoSMonitor) API ... 16
Authority and Locks .................. 16
Required Parameter Group .......... 16
Error Messages ....................... 16
List Saved QoS Monitor Data
(QtoqListSavedQoSMonitorData) API ... 16
Authority and Locks .................. 17
Required Parameter Group .......... 17
QTOQ0100 Format .................... 17
Field Descriptions .................... 18
Error Messages ....................... 18
Save QoS Monitor Data
(QtoqSaveQoSMonitorData) API ....... 18
Authority and Locks .................. 18
Required Parameter Group .......... 18
Error Messages ....................... 19
Start QoS Monitor (QtoqStartQoSMonitor) API ... 19
Authority and Locks .................. 19
Required Parameter Group .......... 19
Error Messages ....................... 20
qtoq_accept()—Accept QoS Sockets Connection API 20
Parameters ......................... 20
Authority .......................... 22
Return Values ....................... 22
Error Conditions ...................... 22
Error Messages ....................... 23
Usage Notes .......................... 23
Related Information ................. 23
qtoq_close()—Close QoS Sockets Connection API ... 23
Parameters ......................... 24
Authority .......................... 24
Return Values ....................... 24
Error Conditions ................. 24
Error Messages ....................... 25
Usage Notes ......................... 25
Related Information ............... 25
qtoq_connect()—Make QoS Sockets Connection API ... 25
Parameters ......................... 25
Authority .......................... 26
Return Values ....................... 27
Error Conditions ................. 27
Error Messages ....................... 29
Usage Notes ......................... 29
Related Information ............... 29
qtoq_ioctl()—Set QoS Sockets Control Options API ... 29
Parameters ......................... 29
Authority .......................... 31
Return Values ....................... 31
Error Conditions ................. 31
Error Messages ....................... 33
Usage Notes ......................... 33
Related Information ............... 33
rapi_dispatch()—Dispatch the RAPI
message-handling routine ............. 33
Parameters ......................... 33
Authority .......................... 34
Return Value ....................... 34
Error Conditions ................. 34
Usage Notes ......................... 34
Related Information ............... 34
rapi_fmt_adspec()—Format a RAPI Adsapc spec ................. 34
Parameters ......................... 35
Authority .......................... 35
Return Value ....................... 35
Error Conditions ................. 35
Usage Notes ......................... 35
Related Information ............... 35
rapi_fmt_filtspec()—Format a RAPI Filter spec ................. 35
Parameters ......................... 36
Authority .......................... 36
Return Value ....................... 36
Error Conditions ................. 36
Usage Notes ......................... 36
Related Information ............... 36
rapi_fmt_flowspec()—Format a RAPI Flowspec spec ................. 36
Parameters ......................... 37
Authority .......................... 37
Return Value ....................... 37
Error Conditions ................. 37
Usage Notes ......................... 37

© Copyright IBM Corp. 1998, 2006
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:

- **Open List of QoS Monitor Data (QgyOpenListQoSMonitorData) API** on page 2 (Open List of QoS Monitor Data) allows the user to gather information related to QoS services.
- **qtoq_accept()—Accept QoS Sockets Connection API** on page 20 (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 API** on page 23 (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 API** on page 25 (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 API** on page 29 (Set QoS Sockets Control Options) provides simplified Quality of Service functionality for connectionless sockets communications between RSVP aware applications on a client and server.
- **Delete QoS Monitor Data (QtoqDeleteQoSMonitorData) API** on page 15 (Delete QoS Monitor Data) allows the user to delete one or more sets of collected QoS monitor data.
- **End QoS Monitor (QtoqEndQoSMonitor) API** on page 16 (End QoS Monitor) allows the user to stop gathering information related to QoS services.
- **List Saved QoS Monitor Data (QtoqListSavedQoSMonitorData) API** on page 16 (List Saved QoS Monitor Data) allows the user to return a list of all collected monitor data that was saved previously.
- **Save QoS Monitor Data (QtoqSaveQoSMonitorData) API** on page 18 (Save QoS Monitor Data) allows the user to save a copy of the collected QoS monitor data for future use.
- **Start QoS Monitor (QtoqStartQoSMonitor) API** on page 19 (Start QoS Monitor) allows the user to gather information related to QoS services.
- **rapi_dispatch()—Dispatch the RAPI message-handling routine** on page 33 (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** on page 34 (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_filspec()—Format a RAPI Filter spec** on page 35 (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 filspec 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” on page 36 (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” on page 37 (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 descriptor to wait on” on page 38 (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” on page 39 (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” on page 40 (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” on page 43 (Identify a RAPI sender) identifies an RSVP sender to potential receivers of the data.

• “rapi_session()—Create a RAPI session” on page 44 (Create a RAPI session) returns an API session ID that is unique to this request.

• “rapi_version()—Retrieve the current RAPI version” on page 46 (Retrieve the current RAPI version) returns the RAPI version currently being used by the RSVP agent.

 APIs

These are the APIs for this category.

Open List of QoS Monitor Data (QgyOpenListQoSMonitorData) API

Required Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Output</th>
<th>Input</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Receiver variable</td>
<td>Char(*)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Length of receiver variable</td>
<td></td>
<td>Binary(4)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>List information</td>
<td></td>
<td>Char(80)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Number of records to return</td>
<td></td>
<td>Binary(4)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Format name</td>
<td></td>
<td>Char(8)</td>
<td></td>
</tr>
</tbody>
</table>

Omissible Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Input</th>
<th>I/O</th>
<th>Char(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Filter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Error code</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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” on page 12.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BIN(4)</td>
<td>Length of filter</td>
</tr>
<tr>
<td>4</td>
<td>BIN(4)</td>
<td>Filter flag</td>
</tr>
<tr>
<td>8</td>
<td>CHAR(14)</td>
<td>Start time</td>
</tr>
<tr>
<td>22</td>
<td>CHAR(14)</td>
<td>End time</td>
</tr>
<tr>
<td>36</td>
<td>BIN(4)</td>
<td>Policy flag</td>
</tr>
<tr>
<td>40</td>
<td>BIN(4)</td>
<td>System aggregation flag</td>
</tr>
<tr>
<td>44</td>
<td>CHAR(128)</td>
<td>Policy name</td>
</tr>
<tr>
<td>172</td>
<td>CHAR(10)</td>
<td>Saved collected name</td>
</tr>
</tbody>
</table>

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 then 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” on page 12.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>CHAR(128)</td>
<td>Policy name</td>
</tr>
<tr>
<td>128</td>
<td>80</td>
<td>CHAR(14)</td>
<td>Time stamp</td>
</tr>
<tr>
<td>142</td>
<td>8E</td>
<td>CHAR(2)</td>
<td>Reserved - alignment</td>
</tr>
<tr>
<td>144</td>
<td>90</td>
<td>BINARY(4)</td>
<td>Protocol</td>
</tr>
<tr>
<td>148</td>
<td>94</td>
<td>CHAR(15)</td>
<td>Source IP address (start) - dotted decimal</td>
</tr>
<tr>
<td>163</td>
<td>A3</td>
<td>CHAR(15)</td>
<td>Destination IP address (start) - dotted decimal</td>
</tr>
<tr>
<td>178</td>
<td>B2</td>
<td>CHAR(15)</td>
<td>Source IP address (end) - dotted decimal</td>
</tr>
<tr>
<td>193</td>
<td>C1</td>
<td>CHAR(15)</td>
<td>Destination IP address (end) - dotted decimal</td>
</tr>
<tr>
<td>208</td>
<td>D0</td>
<td>BINARY(4)</td>
<td>Source port (start)</td>
</tr>
<tr>
<td>212</td>
<td>D4</td>
<td>BINARY(4)</td>
<td>Destination port (start)</td>
</tr>
<tr>
<td>216</td>
<td>D8</td>
<td>BINARY(4)</td>
<td>Source port (end)</td>
</tr>
<tr>
<td>220</td>
<td>DC</td>
<td>BINARY(4)</td>
<td>Destination port (end)</td>
</tr>
<tr>
<td>224</td>
<td>E0</td>
<td>BINARY(4)</td>
<td>Token bucket rate - bytes per second</td>
</tr>
<tr>
<td>228</td>
<td>E4</td>
<td>BINARY(4)</td>
<td>Token bucket depth - bytes</td>
</tr>
<tr>
<td>232</td>
<td>E8</td>
<td>BINARY(4)</td>
<td>Peak data rate - bytes per second</td>
</tr>
<tr>
<td>236</td>
<td>EC</td>
<td>BINARY(4)</td>
<td>Minimum policed unit - bytes</td>
</tr>
<tr>
<td>240</td>
<td>F0</td>
<td>BINARY(4)</td>
<td>Maximum packet size - bytes</td>
</tr>
<tr>
<td>244</td>
<td>F4</td>
<td>BINARY(4)</td>
<td>Total connections serviced - connections</td>
</tr>
<tr>
<td>248</td>
<td>F8</td>
<td>BINARY(4)</td>
<td>Total packets transmitted - packets</td>
</tr>
<tr>
<td>252</td>
<td>FC</td>
<td>BINARY(4)</td>
<td>Total bytes transmitted - bytes</td>
</tr>
<tr>
<td>256</td>
<td>100</td>
<td>BINARY(4)</td>
<td>Total in profile packets - packets</td>
</tr>
<tr>
<td>260</td>
<td>104</td>
<td>BINARY(4)</td>
<td>Total in profile bytes - bytes</td>
</tr>
</tbody>
</table>
**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" on page 12.

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

**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" on page 12.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Returns everything from QOSM0100</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4) Guaranteed rate - bytes per second</td>
</tr>
<tr>
<td>264</td>
<td>108</td>
<td>BINARY(4) Slack term - second</td>
</tr>
</tbody>
</table>
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" on page 12.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4) Guaranteed rate - kbits per second</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4) Slack term - second</td>
</tr>
</tbody>
</table>

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" on page 12.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>CHAR(128) Policy name</td>
</tr>
<tr>
<td>128</td>
<td>80</td>
<td>CHAR(14) Time stamp</td>
</tr>
<tr>
<td>142</td>
<td>8E</td>
<td>CHAR(2) Reserved - alignment</td>
</tr>
<tr>
<td>144</td>
<td>90</td>
<td>BINARY(4) Priority</td>
</tr>
<tr>
<td>148</td>
<td>94</td>
<td>BINARY(4) Protocol</td>
</tr>
<tr>
<td>152</td>
<td>98</td>
<td>CHAR(15) Source IP address (start) - dotted decimal</td>
</tr>
<tr>
<td>167</td>
<td>A7</td>
<td>CHAR(15) Destination IP address (start) - dotted decimal</td>
</tr>
<tr>
<td>182</td>
<td>B6</td>
<td>CHAR(15) Source IP address (end) - dotted decimal</td>
</tr>
<tr>
<td>197</td>
<td>C5</td>
<td>CHAR(15) Destination IP address (end) - dotted decimal</td>
</tr>
<tr>
<td>212</td>
<td>D4</td>
<td>BINARY(4) Source port (start)</td>
</tr>
<tr>
<td>216</td>
<td>D8</td>
<td>BINARY(4) Destination port (start)</td>
</tr>
<tr>
<td>220</td>
<td>DC</td>
<td>BINARY(4) Source port (end)</td>
</tr>
<tr>
<td>224</td>
<td>E0</td>
<td>BINARY(4) Destination port (end)</td>
</tr>
<tr>
<td>228</td>
<td>E4</td>
<td>BINARY(4) Token bucket rate - bytes per second</td>
</tr>
<tr>
<td>232</td>
<td>E8</td>
<td>BINARY(4) Token bucket depth - bytes</td>
</tr>
<tr>
<td>236</td>
<td>EC</td>
<td>BINARY(4) Peak data rate - bytes</td>
</tr>
<tr>
<td>240</td>
<td>F0</td>
<td>CHAR(1) InDSCP</td>
</tr>
<tr>
<td>241</td>
<td>F1</td>
<td>CHAR(1) OutDSCP</td>
</tr>
<tr>
<td>242</td>
<td>F2</td>
<td>CHAR(2) Reserved - alignment</td>
</tr>
<tr>
<td>244</td>
<td>F4</td>
<td>BINARY(4) Total packets transmitted - packets</td>
</tr>
<tr>
<td>248</td>
<td>F8</td>
<td>BINARY(4) Total bytes transmitted - bytes</td>
</tr>
<tr>
<td>252</td>
<td>FC</td>
<td>BINARY(4) Total in profile packets - packets</td>
</tr>
<tr>
<td>256</td>
<td>100</td>
<td>BINARY(4) Total in profile bytes - bytes</td>
</tr>
<tr>
<td>260</td>
<td>104</td>
<td>BINARY(4) Total active connections - connections</td>
</tr>
<tr>
<td>264</td>
<td>108</td>
<td>BINARY(4) Traffic profile</td>
</tr>
</tbody>
</table>
# 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” on page 12.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CHAR(128)</td>
<td>Policy name</td>
</tr>
<tr>
<td>128</td>
<td>CHAR(14)</td>
<td>Time stamp</td>
</tr>
<tr>
<td>142</td>
<td>CHAR(2)</td>
<td>Reserved - alignment</td>
</tr>
<tr>
<td>144</td>
<td>BINARY(4)</td>
<td>Priority</td>
</tr>
<tr>
<td>148</td>
<td>BINARY(4)</td>
<td>Protocol</td>
</tr>
<tr>
<td>152</td>
<td>CHAR(15)</td>
<td>Source IP address (start) - dotted decimal</td>
</tr>
<tr>
<td>167</td>
<td>CHAR(15)</td>
<td>Destination IP address (start) - dotted decimal</td>
</tr>
<tr>
<td>182</td>
<td>CHAR(15)</td>
<td>Source IP address (end) - dotted decimal</td>
</tr>
<tr>
<td>197</td>
<td>CHAR(15)</td>
<td>Destination IP address (end) - dotted decimal</td>
</tr>
<tr>
<td>212</td>
<td>BINARY(4)</td>
<td>Source port (start)</td>
</tr>
<tr>
<td>216</td>
<td>BINARY(4)</td>
<td>Destination port (start)</td>
</tr>
<tr>
<td>220</td>
<td>BINARY(4)</td>
<td>Source port (end)</td>
</tr>
<tr>
<td>224</td>
<td>BINARY(4)</td>
<td>Destination port (end)</td>
</tr>
<tr>
<td>228</td>
<td>BINARY(4)</td>
<td>Token bucket rate - kbits per second</td>
</tr>
<tr>
<td>232</td>
<td>BINARY(4)</td>
<td>Token bucket depth - kbits</td>
</tr>
<tr>
<td>236</td>
<td>BINARY(4)</td>
<td>Peak data rate - kbits per second</td>
</tr>
<tr>
<td>240</td>
<td>CHAR(1)</td>
<td>InDSCP</td>
</tr>
<tr>
<td>241</td>
<td>CHAR(1)</td>
<td>OutDSCP</td>
</tr>
<tr>
<td>242</td>
<td>CHAR(2)</td>
<td>Reserved - alignment</td>
</tr>
<tr>
<td>244</td>
<td>BINARY(8)</td>
<td>Total packets transmitted long - packets</td>
</tr>
<tr>
<td>252</td>
<td>BINARY(8)</td>
<td>Total kbits transmitted long - kbits</td>
</tr>
<tr>
<td>260</td>
<td>BINARY(8)</td>
<td>Total in profile packets long - packets</td>
</tr>
<tr>
<td>268</td>
<td>BINARY(8)</td>
<td>Total in profile kbits long - kbits</td>
</tr>
<tr>
<td>276</td>
<td>BINARY(8)</td>
<td>Total active connections long - connections</td>
</tr>
<tr>
<td>284</td>
<td>BINARY(4)</td>
<td>Traffic profile</td>
</tr>
<tr>
<td>288</td>
<td>BINARY(4)</td>
<td>Duration - seconds</td>
</tr>
<tr>
<td>292</td>
<td>BINARY(4)</td>
<td>Policy handle identifier</td>
</tr>
</tbody>
</table>

# 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” on page 12.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CHAR(128)</td>
<td>Policy name</td>
</tr>
<tr>
<td>128</td>
<td>BINARY(4)</td>
<td>Token bucket rate - bytes per second</td>
</tr>
</tbody>
</table>
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” on page 12.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>132</td>
<td>84</td>
<td></td>
<td>BINARY(4)</td>
<td>Total connects serviced - connections</td>
</tr>
<tr>
<td>136</td>
<td>88</td>
<td></td>
<td>BINARY(4)</td>
<td>Total packets transmitted - packets</td>
</tr>
<tr>
<td>140</td>
<td>8C</td>
<td></td>
<td>BINARY(4)</td>
<td>Total bytes transmitted - bytes</td>
</tr>
<tr>
<td>144</td>
<td>90</td>
<td></td>
<td>BINARY(4)</td>
<td>Total in profile packets - packets</td>
</tr>
<tr>
<td>148</td>
<td>94</td>
<td></td>
<td>BINARY(4)</td>
<td>Total in profile bytes - bytes</td>
</tr>
<tr>
<td>152</td>
<td>98</td>
<td></td>
<td>CHAR(14)</td>
<td>Start time</td>
</tr>
<tr>
<td>166</td>
<td>A6</td>
<td></td>
<td>CHAR(14)</td>
<td>End time</td>
</tr>
</tbody>
</table>

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” on page 12.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>180</td>
<td>B4</td>
<td></td>
<td>BINARY(4)</td>
<td>Returns everything from AGGR0100</td>
</tr>
<tr>
<td>184</td>
<td>B8</td>
<td></td>
<td>BINARY(4)</td>
<td>Guaranteed rate - bytes per second</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BINARY(4)</td>
<td>Actual calculated rate - bytes per second</td>
</tr>
</tbody>
</table>

Resource Reservation Setup Protocol APIs  9
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" on page 12.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
<td>Guaranteed rate - kbits per second</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
<td>Actual calculated rate - kbits per second</td>
</tr>
</tbody>
</table>

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" on page 12.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>CHAR(128)</td>
<td>Policy name</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>80</td>
<td>BINARY(4)</td>
<td>Token bucket rate - bytes per second</td>
</tr>
<tr>
<td></td>
<td>132</td>
<td>84</td>
<td>BINARY(4)</td>
<td>Token bucket depth - bytes</td>
</tr>
<tr>
<td></td>
<td>136</td>
<td>88</td>
<td>CHAR(1)</td>
<td>InDSCP</td>
</tr>
<tr>
<td></td>
<td>137</td>
<td>89</td>
<td>CHAR(1)</td>
<td>OutDSCP</td>
</tr>
<tr>
<td></td>
<td>138</td>
<td>8A</td>
<td>CHAR(2)</td>
<td>Reserved - alignment</td>
</tr>
<tr>
<td></td>
<td>140</td>
<td>8C</td>
<td>BINARY(4)</td>
<td>Total active connections - connections</td>
</tr>
<tr>
<td></td>
<td>144</td>
<td>90</td>
<td>BINARY(4)</td>
<td>Total packets transmitted - packets</td>
</tr>
<tr>
<td></td>
<td>148</td>
<td>94</td>
<td>BINARY(4)</td>
<td>Total bytes transmitted - bytes</td>
</tr>
<tr>
<td></td>
<td>152</td>
<td>98</td>
<td>BINARY(4)</td>
<td>Total in profile packets - packets</td>
</tr>
<tr>
<td></td>
<td>156</td>
<td>9C</td>
<td>BINARY(4)</td>
<td>Total in profile bytes - bytes</td>
</tr>
<tr>
<td></td>
<td>160</td>
<td>A0</td>
<td>BINARY(4)</td>
<td>Traffic profile</td>
</tr>
<tr>
<td></td>
<td>164</td>
<td>A4</td>
<td>CHAR(14)</td>
<td>Start time</td>
</tr>
<tr>
<td></td>
<td>178</td>
<td>B2</td>
<td>CHAR(14)</td>
<td>End time</td>
</tr>
</tbody>
</table>

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" on page 12.

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

**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” on page 12.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>CHAR(128)</td>
<td>Policy name</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>80</td>
<td>CHAR(14)</td>
<td>Time stamp</td>
<td></td>
</tr>
<tr>
<td>142</td>
<td>8E</td>
<td>CHAR(2)</td>
<td>Reserved - alignment</td>
<td></td>
</tr>
<tr>
<td>144</td>
<td>90</td>
<td>BINARY(4)</td>
<td>Priority</td>
<td></td>
</tr>
<tr>
<td>148</td>
<td>94</td>
<td>CHAR(15)</td>
<td>Source IP address (start) - dotted decimal</td>
<td></td>
</tr>
<tr>
<td>163</td>
<td>A3</td>
<td>CHAR(15)</td>
<td>Destination IP address (start) - dotted decimal</td>
<td></td>
</tr>
<tr>
<td>178</td>
<td>B2</td>
<td>CHAR(15)</td>
<td>Source IP address (end) - dotted decimal</td>
<td></td>
</tr>
<tr>
<td>193</td>
<td>C1</td>
<td>CHAR(15)</td>
<td>Destination IP address (end) - dotted decimal</td>
<td></td>
</tr>
<tr>
<td>208</td>
<td>D0</td>
<td>BINARY(4)</td>
<td>Source port (start)</td>
<td></td>
</tr>
<tr>
<td>212</td>
<td>D4</td>
<td>BINARY(4)</td>
<td>Destination port (start)</td>
<td></td>
</tr>
<tr>
<td>216</td>
<td>D8</td>
<td>BINARY(4)</td>
<td>Source port (end)</td>
<td></td>
</tr>
<tr>
<td>220</td>
<td>DC</td>
<td>BINARY(4)</td>
<td>Destination port (end)</td>
<td></td>
</tr>
<tr>
<td>224</td>
<td>E0</td>
<td>BINARY(4)</td>
<td>Average connection rate - connections per second</td>
<td></td>
</tr>
<tr>
<td>228</td>
<td>E4</td>
<td>BINARY(4)</td>
<td>Connection burst - connections</td>
<td></td>
</tr>
<tr>
<td>232</td>
<td>E8</td>
<td>BINARY(4)</td>
<td>Peak connection rate - connections per second</td>
<td></td>
</tr>
<tr>
<td>236</td>
<td>EC</td>
<td>BINARY(4)</td>
<td>Prioritized queue</td>
<td></td>
</tr>
<tr>
<td>240</td>
<td>F0</td>
<td>BINARY(8)</td>
<td>Total connections transmitted - connections</td>
<td></td>
</tr>
<tr>
<td>248</td>
<td>F8</td>
<td>BINARY(8)</td>
<td>Total in profile connections - connections</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>100</td>
<td>BINARY(4)</td>
<td>Duration - seconds</td>
<td></td>
</tr>
<tr>
<td>260</td>
<td>104</td>
<td>BINARY(4)</td>
<td>Policy handle identifier</td>
<td></td>
</tr>
</tbody>
</table>
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."

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHAR(128)</td>
<td>Policy name</td>
</tr>
<tr>
<td>128</td>
<td>80</td>
<td>8E</td>
<td>CHAR(14)</td>
<td>Time stamp</td>
</tr>
<tr>
<td>142</td>
<td>8E</td>
<td>90</td>
<td>CHAR(2)</td>
<td>Reserved - alignment</td>
</tr>
<tr>
<td>144</td>
<td>90</td>
<td>A3</td>
<td>BINARY(4)</td>
<td>Priority</td>
</tr>
<tr>
<td>148</td>
<td>94</td>
<td>94</td>
<td>CHAR(15)</td>
<td>Destination IP address (start) - dotted decimal</td>
</tr>
<tr>
<td>163</td>
<td>A3</td>
<td>C0</td>
<td>CHAR(15)</td>
<td>Destination IP address (end) - dotted decimal</td>
</tr>
<tr>
<td>178</td>
<td>B2</td>
<td>C0</td>
<td>CHAR(2)</td>
<td>Reserved - alignment</td>
</tr>
<tr>
<td>180</td>
<td>B4</td>
<td>C4</td>
<td>BINARY(4)</td>
<td>Destination port (start)</td>
</tr>
<tr>
<td>184</td>
<td>B8</td>
<td>BC</td>
<td>BINARY(4)</td>
<td>Average URI rate - URIs per second</td>
</tr>
<tr>
<td>188</td>
<td>BC</td>
<td>C0</td>
<td>BINARY(4)</td>
<td>URI burst - number of URIs</td>
</tr>
<tr>
<td>192</td>
<td>C0</td>
<td>C8</td>
<td>BINARY(4)</td>
<td>Peak URI rate - URIs per second</td>
</tr>
<tr>
<td>196</td>
<td>C4</td>
<td>C8</td>
<td>BINARY(4)</td>
<td>Prioritized queue</td>
</tr>
<tr>
<td>200</td>
<td>C8</td>
<td>D0</td>
<td>BINARY(8)</td>
<td>Total URIs transmitted - number of URIs</td>
</tr>
<tr>
<td>208</td>
<td>D0</td>
<td>D8</td>
<td>BINARY(4)</td>
<td>Total in profile URIs - number of URIs</td>
</tr>
<tr>
<td>216</td>
<td>D8</td>
<td>DC</td>
<td>BINARY(4)</td>
<td>Duration - seconds</td>
</tr>
<tr>
<td>220</td>
<td>DC</td>
<td>E0</td>
<td>BINARY(4)</td>
<td>Policy handle identifier</td>
</tr>
<tr>
<td>224</td>
<td>E0</td>
<td>CHAR(128)</td>
<td>URI name</td>
<td></td>
</tr>
</tbody>
</table>

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 then 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 then 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 then 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 then 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 then 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

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

API introduced: V5R1

Delete QoS Monitor Data (QtoqDeleteQoSMonitorData) API

Required Parameter Group:

1. QoS collection name or names
2. Length of QoS collection name or names
3. Error code

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

End QoS Monitor (QtoqEndQoS Monitor) API

Required Parameter Group:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Error Code</td>
<td>I/O</td>
</tr>
</tbody>
</table>

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

The End QoS Monitor (QtoqEndQoS Monitor) 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.

API introduced: V5R1

List Saved QoS Monitor Data (QtoqListSavedQoS Monitor Data) API

Required Parameter Group:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Receiver variable</td>
<td>Output</td>
</tr>
</tbody>
</table>
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 the 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 contain QoS Monitor Data.

**Error code**

**I/O; CHAR(*)**

The structure in which to return error information. For the format of the structure, see the 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 the Field Descriptions.
### 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

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3C1E E</td>
<td>Required parameter &amp;1 omitted.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF9810 E</td>
<td>Library &amp;1 not found.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V5R2

---

### Save QoS Monitor Data (QtoqSaveQoSMonitorData) API

**Required Parameter Group:**

- **QoS collection name**
  - Output
  - Char(10)
- **Description text**
  - Input
  - Char(50)
- **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

Start QoS Monitor (QtoqStartQoSMonitor) API

Required Parameter Group:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Wrap</td>
<td>Input</td>
</tr>
<tr>
<td>2</td>
<td>Buffer size</td>
<td>Input</td>
</tr>
<tr>
<td>3</td>
<td>Granularity</td>
<td>Input</td>
</tr>
<tr>
<td>4</td>
<td>Error Code</td>
<td>I/O</td>
</tr>
</tbody>
</table>

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:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not wrap the buffer</td>
</tr>
<tr>
<td>1</td>
<td>Wrap the buffer</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP9215 E</td>
<td>QoS Monitor is active (not a valid state).</td>
</tr>
<tr>
<td>CPF24B4 E</td>
<td>Severe error addressing parameter list.</td>
</tr>
<tr>
<td>CPF3C1E E</td>
<td>Required parameter &amp;1 omitted.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of &amp;1 API.</td>
</tr>
</tbody>
</table>

API introduced: V5R1

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:

- **REQ_SIGNAL_RET_EVENTS (1)**: 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.
- **REQ_NOSIGNAL (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 microsecs */
        } Guar_spec;
    } spec_u;
    qos_spec_t Spec;  /* Flow info */
    unsigned char result; /* API status */
} qos_req;  /* End of QoS request structure */

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_session;  /* End of QoS request structure */

Resource Reservation Setup Protocol APIs 21
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 qtq_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.
There is not enough buffer space for the requested operation.

The specified descriptor does not reference a socket.

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).

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

Unknown system state.

Operation would have caused the thread to be suspended.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

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

Syntax

```c
#include <qtoqsapi.h>

int qtoq_close(
```
### 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.

- **[ENOBUFFS]**  
  There is not enough buffer space for the requested operation.

- **[EUNKNOWN]**  
  Unknown system state.
Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

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

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:

REQ_SIGNAL_RET_EVENTS(1) 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 microsecs */
        } 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.
An underlying protocol error has occurred.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

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

```
#include <qtoqsapi.h>

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

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:

- `REQ_SIGNAL_RET_EVENTS(1)`: 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.
- `REQ_NOSIGNAL(3)`: Load specified QoS policy if admission control allows it.
- `REQ_GET_RSVP_DATA(4)`: Get the RSVP flowspec that has been returned as the result of an RSVP event. This request is valid only if a previous `REQ_SIGNAL_RET_EVENTS` 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:

```c
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 microsecs */
        } 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.

[ENOMEM]
There is not enough buffer space for the requested operation.

[ENOTDIR]
Not a directory.

[EOPNOTSUPP]
Operation not supported.

[ETIMEOUT]
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().
Unknown system state.

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

An underlying protocol error has occurred.

### Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPE3418 E</td>
<td>Possible APAR condition or hardware failure.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
<tr>
<td>CPFA081 E</td>
<td>Unable to set return value or error code.</td>
</tr>
</tbody>
</table>

### 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

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

**Syntax**

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

**Service program name**: QSYS/QTOQRAPI

**Default public authority**: *EXCLUDE

**Threaddisable**: 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_INVAL]
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

rapi_fmt_adspec()—Format a RAPI Adspec

Syntax
#include <rapi.h>
void rapi_fmt_adspec(
    rapi_adspec_t *pAdspec,
    char *pBuffer,
    int size
)

Service program name: QSYS/QTOQRAPI
Default public authority: *EXCLUDE
Threadsafe: Yes

The rapi_fmt_adspec() API 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. The
Adspec 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**

\[ p\text{Adspec} \]
(Input) Required
A pointer to the Adspec to be formatted.

\[ p\text{Buffer} \]
(Input/Output) Required
A pointer to the buffer to be used.

\[ \text{size} \]
(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

---

**rapi_fmt_filtspec()—Format a RAPI Filter spec**

**Syntax**

```c
#include <rapi.h>
void rapi_fmt_filtspec(
    rapi_filter_t *pFiltspec,
    char *pBuffer,
    int size )
```

Service program name: QSYS/QTOQRAPI
Default public authority: *EXCLUDE
Threadsafe: Yes
The `rapi_fmt_filtspec()` API 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. The filtspec 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

---

**rapi_fmt_flowspec()—Format a RAPI Flowspec**

**Syntax**

```c
#include <rapi.h>

void rapi_fmt_flowspec(
    rapi_flowspec_t *pFlowspec,
    char *pBuffer,
    int size )
```

Service program name: QSYS/QTOQRAPI
Default public authority: *EXCLUDE
Threadsafe: Yes
The `rapi_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

---

**rapi_fmt_tspec()—Format a RAPI Tspec**

**Syntax**

```c
#include <rapi.h>
void rapi_fmt_tspec(
    rapi_tspec_t *pTspec,
    char *pBuffer,
    int size )
```

Service program name: QSYS/QTOQRAPI  
Default public authority: *EXCLUDE  
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

---

**rapi_getfd()—Get descriptor to wait on**

**Syntax**

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

Service program name: QSYS/QTOQRAPI

Default public authority: *EXCLUDE

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

---

**rapi_release()—Release the currently active RAPI reservation**

**Syntax**

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

Service program name: QSYS/QTOQRAPI  
Default public authority: *EXCLUDE  
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

---

**rapi_reserve()**—Make, modify, or delete a RAPI reservation

Syntax

```c
#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 *RcvPoI,
    int numFilt,
    rapi_filter_t *FspecLst,
    int numFlow,
    rapi_flowspec_t *Flowlst )
```
The RSVP receiver uses 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

<table>
<thead>
<tr>
<th>Style Type</th>
<th>Style ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wildcard Filter(WF)</td>
<td>RAPI_RSTYLE_WILDCARD</td>
<td>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).</td>
</tr>
<tr>
<td>Fixed Filter(FF)</td>
<td>RAPI_RSTYLE_FIXED</td>
<td>FilterSpecNo must equal FlowspecNo. Entries Flowspec_list and FilterSpec_list parameters will correspond in pairs.</td>
</tr>
<tr>
<td>Shared Explicit(SE)</td>
<td>_RSTYLE_SE</td>
<td>The Flowspec_list parameter should point to a single flowspec. The FilterSpec_list parameter may point to a list of any length.</td>
</tr>
</tbody>
</table>

### Authorities

None.

### Return Value

Returns 0 if successful.

RAPI error code if it fails.

### Error Conditions

[RAPI_ERR_INVAL]

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
**rapi_sender()**—Identify a RAPI sender

Syntax

```c
#include <rapi.h>
int rapi_sender (
    rapi_sid_t SessID,
    int flags,
    rapi_addr_t *LocalAddr,
    rapi_filter_t *Filter,
    rapi_tspec_t *SndTspec,
    rapi_adspec_t *SndAdspec,
    rapi_policy_t *SndPol,
    int Ttl )
```

Service program name: QSYS/QTOQRAPI
Default public authority: *EXCLUDE
Threadsafe: Yes

The `rapi_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 `rapi_session` call.

**Parameters**

**SessID**

(Input) Required  
The session ID returned by a successful `rapi_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_INVAL*  
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 APIs.
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

---

**rapi_session()**—Create a RAPI session

**Syntax**

```c
#include <rapi.h>

rapi_sid_t rapi_session(
    rapi_addr_t  *Dest,
    int    Protid,

rapi_addr_t *Dest,
int Protid,
```
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 is 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_INVAL]

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](http://www.ietf.org).

Complete documentation of the RAPI APIs can be found at [The Open Group](http://www.opengroup.org).

API introduced: V5R1

---

**rapi_version()—Retrieve the current RAPI version**

Syntax

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

Service program name: QSYS/QTOQRAPI

Default public authority: *EXCLUDE

Threading: 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
Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:
Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(C) IBM 2006. Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. 1998, 2006. All rights reserved.

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

**Programming Interface Information**

This Application Programming Interfaces (API) publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.
Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

Advanced 36
Advanced Function Printing
Advanced Peer-to-Peer Networking
AFP
AIX
AS/400
COBOL/400
CUA
DB2
DB2 Universal Database
Distributed Relational Database Architecture
Domino
DPI
DRDA
eServer
GDDM
IBM
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
i5/OS
iSeries
Lotus Notes
MVS
Netfinity
Net.Data
NetView
Notes
OfficeVision
Operating System /2
Operating System/400
OS/2
OS/400
PartnerWorld
PowerPC
PrintManager
Print Services Facility
RISC System/6000
RPG/400
RS/6000
SAA
SecureWay
System/36
System/370
System/38
System/390
VisualAge
WebSphere
xSeries

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Terms and Conditions

Permissions for the use of these Publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these Publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these Publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these Publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these Publications, or reproduce, distribute or display these Publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the Publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations. IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.