



@server

iSeries

Communications APIs

*Version 5 Release 3*







@server

iSeries

Communications APIs

*Version 5 Release 3*

**Note**

Before using this information and the product it supports, be sure to read the information in "Notices," on page 305.

**Sixth Edition (August 2005)**

This edition applies to version 5, release 3, modification 0 of Operating System/400 (product number 5722-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

© Copyright International Business Machines Corporation 1998, 2005. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Communications APIs</b> . . . . .	<b>1</b>	Close Path (QzdmClosePath) API . . . . .	80
APIs . . . . .	1	Restrictions . . . . .	81
User-Defined Communications Support APIs . . . . .	1	Authority and Locks . . . . .	81
Disable Link (QOLDLINK) API . . . . .	2	Required Parameter Group . . . . .	81
Required Parameter Group . . . . .	2	CPTH0100 Format . . . . .	81
Return and Reason Codes . . . . .	3	Field Descriptions . . . . .	81
Enable Link (QOLELINK) API . . . . .	4	Error Messages . . . . .	82
Authorities and Locks . . . . .	5	Close Stream (QzdmCloseStream) API . . . . .	82
Required Parameter Group . . . . .	5	Restrictions . . . . .	82
Optional Parameter Group . . . . .	8	Authority and Locks . . . . .	83
Return and Reason Codes . . . . .	8	Required Parameter Group . . . . .	83
Error Messages . . . . .	9	CSTR0100 Format . . . . .	83
Query Line Description (QOLQLIND) API . . . . .	10	Field Descriptions . . . . .	83
Required Parameter Group . . . . .	10	Error Messages . . . . .	83
Optional Parameter Group . . . . .	11	Open Path (QzdmOpenPath) API . . . . .	84
Format of Data in the User Buffer . . . . .	12	Restrictions . . . . .	84
Return and Reason Codes . . . . .	20	Authority and Locks . . . . .	85
Error Messages . . . . .	21	Required Parameter Group . . . . .	85
Receive Data (QOLRECV) API . . . . .	21	OPRC0100 Format . . . . .	85
Required Parameter Group . . . . .	22	OPRQ0100 Format . . . . .	85
Format of Diagnostic Data Parameter . . . . .	23	Field Descriptions . . . . .	86
LAN Input Operations . . . . .	25	Error Messages . . . . .	86
X.25 SVC and PVC Input Operations . . . . .	27	Open Stream (QzdmOpenStream) API . . . . .	87
Return and Reason Codes . . . . .	34	Restrictions . . . . .	87
Error Messages . . . . .	37	Authority and Locks . . . . .	87
Send Data (QOLSEND) API . . . . .	38	Required Parameter Group . . . . .	87
Required Parameter Group . . . . .	38	OSTR0100 Format . . . . .	88
Diagnostic Data Parameter Format . . . . .	40	Field Descriptions . . . . .	88
LAN Output Operations . . . . .	42	Error Messages . . . . .	88
X.25 SVC and PVC Output Operations . . . . .	43	Receive Control (QzdmReceiveControl) API . . . . .	89
Return and Reason Codes . . . . .	56	Restrictions . . . . .	89
Error Messages . . . . .	59	Authority and Locks . . . . .	89
Set Filter (QOLSETF) API . . . . .	60	Required Parameter Group . . . . .	90
Required Parameter Group . . . . .	61	RCRC0100 Format . . . . .	90
Format of Filter Information . . . . .	61	RCRQ0100 Format . . . . .	90
Return and Reason Codes . . . . .	67	Field Descriptions . . . . .	91
Error Messages . . . . .	68	Error Messages . . . . .	91
Set Timer (QOLTIMER) API . . . . .	69	Receive Request (QzdmReceiveRequest) API . . . . .	92
Required Parameter Group . . . . .	70	Restrictions . . . . .	92
Optional Parameter . . . . .	71	Authority and Locks . . . . .	92
Return and Reason Codes . . . . .	71	Required Parameter Group . . . . .	93
Error Messages . . . . .	72	RQRC0100 Format . . . . .	93
Data Stream Translation APIs . . . . .	72	RQRQ0100 Format . . . . .	93
End Data Stream Translation Session (QD0ENDTS) API . . . . .	72	Field Descriptions . . . . .	94
Required Parameter Group . . . . .	73	Error Messages . . . . .	95
Error Messages . . . . .	73	Receive Response (QzdmReceiveResponse) API . . . . .	96
Start Data Stream Translation Session (QD0STRTS) API . . . . .	73	Restrictions . . . . .	96
Authorities and Locks . . . . .	73	Authority and Locks . . . . .	97
Required Parameter Group . . . . .	74	Required Parameter Group . . . . .	97
Error Messages . . . . .	75	RSRC0100 Format . . . . .	97
Translate Data Stream (QD0TRNDS) API . . . . .	76	RSRQ0100 Format . . . . .	98
Required Parameter Group . . . . .	76	Field Descriptions . . . . .	98
Error Messages . . . . .	78	Error Messages . . . . .	98
OptiConnect APIs . . . . .	79	Send Request (QzdmSendRequest) API . . . . .	99
		Restrictions . . . . .	99
		Authority and Locks . . . . .	100

Required Parameter Group . . . . .	100	NIFC0100 Format . . . . .	128
SRRC0100 Format . . . . .	100	Field Descriptions . . . . .	129
SRRQ0100 Format . . . . .	101	NIFC0200 Format . . . . .	133
Field Descriptions . . . . .	101	Field Descriptions . . . . .	134
Error Messages . . . . .	102	Error Messages . . . . .	138
Send Response (QzdmSendResponse) API . . . . .	102	List Network Routes (QtocLstNetRte) API . . . . .	139
Restrictions . . . . .	103	Authorities and Locks . . . . .	139
Authority and Locks . . . . .	103	Required Parameter Group . . . . .	139
Required Parameter Group . . . . .	103	Format of Route Lists . . . . .	140
SRSP0100 Format . . . . .	103	Input Parameter Section . . . . .	140
Field Descriptions . . . . .	104	Header Section . . . . .	141
Error Messages . . . . .	104	Format of Returned Connection Data . . . . .	141
Wait Message (QzdmWaitMessage) API . . . . .	105	NRTE0100 Format . . . . .	141
Restrictions . . . . .	105	Field Descriptions . . . . .	142
Authority and Locks . . . . .	106	NRTE0200 Format . . . . .	145
Required Parameter Group . . . . .	106	Field Descriptions . . . . .	145
WMRC0100 Format . . . . .	106	Error Messages . . . . .	150
WMRQ0100 Format . . . . .	106	List Physical Interface ARP Table	
Field Descriptions . . . . .	107	(QtocLstPhyIfcARPTbl) API . . . . .	150
Error Messages . . . . .	107	Authorities and Locks . . . . .	151
TCP/IP Management . . . . .	108	Required Parameter Group . . . . .	151
Change Connection Attribute (QTOCCNA) API . . . . .	109	Format of ARP Table Lists . . . . .	151
Authorities and Locks . . . . .	109	Input Parameter Section . . . . .	152
Required Parameter Group . . . . .	109	Header Section . . . . .	152
TCPA0001 Format . . . . .	110	ARPT0100 Format . . . . .	152
UDPA0001 Format . . . . .	110	Field Descriptions . . . . .	152
Field Descriptions . . . . .	110	Error Messages . . . . .	153
Error Messages . . . . .	111	List Physical Interface Data (QtocLstPhyIfcDta) API . . . . .	154
List Neighbor Cache Table (QtocLstNeighborTbl)		Authorities and Locks . . . . .	154
API . . . . .	111	Required Parameter Group . . . . .	154
Authorities and Locks . . . . .	111	Format of Physical Interface Lists . . . . .	155
Required Parameter Group . . . . .	112	Input Parameter Section . . . . .	155
Format of Neighbor Cache Table Lists . . . . .	112	Header Section . . . . .	155
Input Parameter Section . . . . .	113	Format of Returned Connection Data . . . . .	156
Header Section . . . . .	113	IFCD0100 Format . . . . .	156
NNCT0100 Format . . . . .	113	Field Descriptions . . . . .	156
Field Descriptions . . . . .	114	IFCD0200 Format . . . . .	159
Error Messages . . . . .	115	Field Descriptions . . . . .	160
List Network Connections (QtocLstNetCnn) API . . . . .	116	IFCD0300 Format . . . . .	162
Authorities and Locks . . . . .	117	Field Descriptions . . . . .	163
Required Parameter Group . . . . .	117	Error Messages . . . . .	167
Format of Connection Status Lists . . . . .	117	List PPP Connection Profiles (QtocLstPPPCnnPrf)	
Input Parameter Section . . . . .	118	API . . . . .	167
Header Section . . . . .	118	Authorities and Locks . . . . .	168
NCLQ0100 Format . . . . .	118	Required Parameter Group . . . . .	168
Field Descriptions . . . . .	119	Format of Connection Profile Lists . . . . .	168
NCLQ0200 Format . . . . .	120	Input Parameter Section . . . . .	169
Field Descriptions . . . . .	120	Header Section . . . . .	169
Format of Returned Connection Data . . . . .	122	PRFD0100 Format . . . . .	169
NCNN0100 Format . . . . .	122	Field Descriptions . . . . .	169
Field Descriptions . . . . .	122	Error Messages . . . . .	172
NCNN0200 Format . . . . .	124	List TCP/IP Point-to-Point Jobs (QTOCLPPJ) API . . . . .	172
Field Descriptions . . . . .	124	Authorities and Locks . . . . .	172
Error Messages . . . . .	126	Required Parameter Group . . . . .	172
List Network Interfaces (QtocLstNetIfc) API . . . . .	126	Format of Point-to-Point Jobs List . . . . .	173
Authorities and Locks . . . . .	127	Input Parameter Section . . . . .	173
Required Parameter Group . . . . .	127	Header Section . . . . .	174
Format of Interface Lists . . . . .	127	PPPJ0100 Format . . . . .	174
Input Parameter Section . . . . .	128	Field Descriptions . . . . .	174
Header Section . . . . .	128	Error Messages . . . . .	175
Format of Returned Connection Data . . . . .	128	Remove ARP Table Entry (QtocRmvARPTblE) API . . . . .	175

Authorities and Locks . . . . .	175	DNSA0100 Format . . . . .	232
Required Parameter Group . . . . .	176	Field Descriptions . . . . .	232
Error Messages . . . . .	176	Error Messages . . . . .	233
Retrieve Network Connection Data		CPI Communications (CPI-C) . . . . .	233
(QtocRtvNetCnnDta) API . . . . .	177	Exit Programs . . . . .	233
Authorities and Locks . . . . .	177	Trace Exit Program for Trace TCP Application	
Required Parameter Group . . . . .	177	command . . . . .	234
Socket Connection Request Format . . . . .	178	Required Parameter Group . . . . .	234
IPv4 connection (Protocol field value is 1 or 2)	178	Field Descriptions . . . . .	236
IPv6 connection (Protocol field value is 3 or 4)	178	Exit Program for Watch for Trace Event . . . . .	237
Field Descriptions . . . . .	178	Required Parameter Group . . . . .	237
Format of Returned Connection Data . . . . .	179	Field Descriptions . . . . .	238
NCND0100 Format . . . . .	179	Concepts . . . . .	239
Field Descriptions . . . . .	180	User-Defined Communications . . . . .	239
NCND0200 Format . . . . .	181	Overview . . . . .	239
List of Socket Options . . . . .	182	User-Defined Communications Callable	
List of Jobs/Tasks Associated with this		Routines . . . . .	240
Connection . . . . .	182	Input/Output Buffers and Descriptors . . . . .	241
Field Descriptions . . . . .	183	Queues . . . . .	241
NCND1100 Format . . . . .	187	Terminology . . . . .	241
Field Descriptions . . . . .	188	Relationship to Communications Standards . . . . .	242
NCND1200 Format . . . . .	189	Local Area Network (LAN) Considerations . . . . .	245
List of Socket Options . . . . .	190	X.25 Considerations . . . . .	246
List of Jobs/Tasks Associated with this		Programming Design Considerations for	
Connection . . . . .	190	Communications APIs . . . . .	247
Field Descriptions . . . . .	191	Jobs . . . . .	247
Error Messages . . . . .	195	Application Program Feedback . . . . .	250
Retrieve PPP Connection Profiles		Synchronous and Asynchronous Operations . . . . .	250
(QtocRtvPPPCnnPrf) API . . . . .	196	Programming Languages . . . . .	250
Authorities and Locks . . . . .	196	Starting and Ending Communications . . . . .	250
Required Parameter Group . . . . .	196	Using Connection Identifiers . . . . .	251
Format of Connection Profile Attributes		Incoming Connections . . . . .	262
Information . . . . .	197	Closing Connections . . . . .	267
PRFR0100 Format . . . . .	197	Programming Considerations for LAN	
Field Descriptions . . . . .	198	Applications . . . . .	268
PRFR0200 Format . . . . .	199	Operations . . . . .	269
Field Descriptions . . . . .	201	Configuration . . . . .	269
Connection Profile Detailed Parameters . . . . .	208	Inbound Routing Information . . . . .	269
Field Descriptions . . . . .	209	End-to-End Connectivity . . . . .	269
Remote Phone Numbers . . . . .	211	Sending and Receiving Data . . . . .	270
Field Descriptions . . . . .	211	Maximum Amount of Outstanding Data . . . . .	270
Error Messages . . . . .	212	Ethernet to Token-Ring Conversion and Routing	270
Retrieve TCP/IP Attributes (QtocRtvTCPA) API	212	Performance Considerations . . . . .	270
Authorities and Locks . . . . .	212	Programming Considerations for X.25 Applications	271
Required Parameter Group . . . . .	212	X.25 Packet Types Supported . . . . .	271
Format of TCP/IP Attributes Information . . . . .	213	Operations . . . . .	272
TCPA0100 Format . . . . .	213	Connections . . . . .	272
Field Descriptions . . . . .	214	Connection Identifiers . . . . .	273
TCPA0200 Format . . . . .	216	Connection Information . . . . .	273
Field Descriptions . . . . .	216	Switched Virtual Circuit (SVC) Connectivity . . . . .	274
TCPA0300 Format . . . . .	221	Inbound Routing Information . . . . .	274
Field Descriptions . . . . .	222	End-to-End Connectivity . . . . .	274
TCPA1100 Format . . . . .	223	Permanent Virtual Circuit (PVC) Connectivity	274
Field Descriptions . . . . .	223	Inbound Routing Information . . . . .	275
TCPA1200 Format . . . . .	224	End-to-End Connectivity . . . . .	275
Field Descriptions . . . . .	224	Sending and Receiving Data Packets . . . . .	275
Error Messages . . . . .	227	X.25 Call Control . . . . .	277
Update DNS API (QTOBUPDT) . . . . .	228	Performance Considerations . . . . .	277
Authorities and Locks . . . . .	229	Queue Considerations . . . . .	277
Required Parameter Group . . . . .	229	User Space Considerations . . . . .	281
Update Instructions Syntax . . . . .	231	Return Codes and Reason Codes . . . . .	283

Messages . . . . .	283	Display Job Log . . . . .	290
Configuration and Queue Entries. . . . .	284	Display Connection Status . . . . .	290
Configuring User-Defined Communications		Display Inbound Routing Information . . . . .	290
Support . . . . .	284	Work with Communications Trace . . . . .	290
Links . . . . .	284	Work with Error Log . . . . .	290
Queue. . . . .	285	Dump System Object to View User Spaces. . . . .	291
Queue Entries . . . . .	285	Error Codes . . . . .	299
General Format. . . . .	285	Local Area Network (LAN) Error Codes . . . . .	299
Enable-Complete Entry . . . . .	286	X.25 Error Codes . . . . .	300
Disable-Complete Entry . . . . .	287	Common Errors and Messages . . . . .	302
Permanent-Link-Failure Entry . . . . .	287		
Incoming-Data Entry . . . . .	288		
Timer-Expired Entry . . . . .	288		
Debugging of User-Defined Communications			
Applications. . . . .	289	<b>Appendix. Notices . . . . .</b>	<b>305</b>
System Services and Tools . . . . .	289	Trademarks . . . . .	306
Program Debug . . . . .	289	Terms and conditions for downloading and	
Work with Communications Status . . . . .	289	printing publications . . . . .	307
		Code disclaimer information . . . . .	308



---

## Communications APIs

The Communications APIs provide the information needed to write user-defined communications applications, programming examples, and debugging information. The Data Stream Translation APIs allow a user-written application program that creates 3270 data streams to run on the an iSeries server using 5250 data streams. The OptiConnect APIs can be used to move user data between two or more systems that are connected by an OptiConnect fiber-optic bus. The TCP/IP Management APIs allow you to retrieve information about your TCP/IP setup and status, and change certain system values related to TCP/IP.

Communications APIs include the following:

- “User-Defined Communications Support APIs”
- “Data Stream Translation APIs” on page 72
- “OptiConnect APIs” on page 79
- “TCP/IP Management” on page 108
- [»](#) “CPI Communications (CPI-C)” on page 233 [«](#)

For information on user-defined communications support, read the following topics:

- “User-Defined Communications” on page 239
- “Programming Design Considerations for Communications APIs” on page 247
- “Configuration and Queue Entries” on page 284
- “Debugging of User-Defined Communications Applications” on page 289

APIs by category

---

## APIs

These are the APIs for this category.

---

### User-Defined Communications Support APIs

User-defined communications support is made up of seven callable APIs that provide services for a user-defined communications application program.

The user-defined communications Support APIs are:

- “Disable Link (QOLDLINK) API” on page 2 (QOLDLINK) disables one or all links.
- “Enable Link (QOLELINK) API” on page 4 (QOLELINK) enables link for input and output.
- “Query Line Description (QOLQLIND) API” on page 10 (QOLQLIND) queries an existing line description.
- “Receive Data (QOLRECV) API” on page 21 (QOLRECV) receives data from the link.
- “Send Data (QOLSEND) API” on page 38 (QOLSEND) sends data from the link.
- “Set Filter (QOLSETF) API” on page 60 (QOLSETF) activates or deactivates filters.
- “Set Timer (QOLTIMER) API” on page 69 (QOLTIMER) sets or cancels a timer.

Top | “Communications APIs” | APIs by category

---

## Disable Link (QOLDLINK) API

Required Parameter Group:	
<b>1</b>	Return code
<b>Output</b>	Binary(4)
<b>2</b>	Reason code
<b>Output</b>	Binary(4)
<b>3</b>	Communications handle
<b>Input</b>	Char(10)
<b>4</b>	Vary option
<b>Input</b>	Char(1)
	Default Public Authority: *USE
Threadsafe: No	

The Disable Link (QOLDLINK) API disables one or all links that are currently enabled in the job in which the application program is running. When a link is disabled, all system resources that the link is using are released, the input and output buffers and descriptors for that link are deleted, and input or output on that link is no longer possible.

In addition to an application program explicitly disabling a link by calling the QOLDLINK API, user-defined communications support will implicitly disable a link in the following cases:

- When the network device associated with an enabled link is varied off from the job in which it was enabled
- When a job ends in which one or more links were enabled
- When the application program that enabled the link ends abnormally
- When the Reclaim Resource (RCLRSC) command is used
- When an unmonitored escape message is received

For each link that is successfully disabled, either explicitly or implicitly, the disable-complete entry will be sent to the data queue or user queue specified on the call to the QOLELINK API when the link was enabled. See “Disable-Complete Entry” on page 287 for the format of the disable-complete entry.

### Required Parameter Group

#### Return code

OUTPUT; BINARY(4)

The recovery action to take. See “Return and Reason Codes” on page 3.

#### Reason code

OUTPUT; BINARY(4)

The error that occurred. See “Return and Reason Codes” on page 3.

#### Communications handle

INPUT; CHAR(10)

The name of the link to disable. The special value of \*ALL (left-justified and padded on the right with spaces) may be used to disable all links currently enabled in the job that the application program is running in.

## Vary option

INPUT; CHAR(1)

The vary option for the network device description associated with each link being disabled. The valid values are as follows:

*X'00'* Do not vary off the network device description.

*X'01'* Vary off the network device description.

## Return and Reason Codes

### Return and Reason Codes for the QOLDLINK API

Return / Reason Code	Meaning	Recovery
0/0	Operation successful.	Continue processing.
83/1004	Vary option not valid.	Correct the vary option parameter. Then, try the request again.
83/3001	Link not enabled.	Correct the communications handle parameter. Then, try the request again.

API introduced: V2R1

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

## Enable Link (QOLELINK) API

### Required Parameter Group:

- 1        Return code  
**Output**    Binary(4)
- 2        Reason code  
**Output**    Binary(4)
- 3        Data unit size  
**Output**    Binary(4)
- 4        Data units created  
**Output**    Binary(4)
- 5        LAN user data size  
**Output**    Binary(4)
- 6        X.25 data unit size  
**Input**     Binary(4)
- 7        Input buffer  
**Input**     Char(20)
- 8        Input buffer descriptor  
**Input**     Char(20)
- 9        Output buffer  
**Input**     Char(20)
- 10       Output buffer descriptor  
**Input**     Char(20)
- 11       Key length  
**Input**     Binary(4)
- 12       Key value  
**Input**     Char(256)
- 13       Qualified queue name  
**Input**     Char(20)
- 14       Line description  
**Input**     Char(10)
- 15       Communications handle  
**Input**     Char(10)

### Optional Parameter Group:

- 16       Queue type  
**Input**     Char(1)
- 17       Network interface description  
**Input**     Char(10)
- 18       Extended operations  
**Input**     Char(1)

Default Public Authority: \*USE

The Enable Link (QOLELINK) API enables a link for input and output on a communications line. The communications line, described by the line description parameter, must be a token-ring, Ethernet, wireless, FDDI, or X.25 line. The link being enabled can only be accessed within the job in which the QOLELINK API was called.


Before calling the QOLELINK API to enable a link, you must configure the following objects:

- Token-ring, Ethernet, wireless, FDDI, or X.25 line description
- Data queue or user queue
- Network interface description for X.25 networks running over ISDN

See for more information on configuration.

The QOLELINK API creates the input and output buffers and buffer descriptors used for the link being enabled. The network controller description and the network device description, associated with the link being enabled, are also created, if necessary. In addition, the following are varied on, if necessary.

- Line description
- Network controller description
- Network device description
- Network interface descriptions used by the line description

If the X.25 switched network interface list has multiple network interface descriptions configured, all of them can be varied on at one time. For more information on varying on network interface descriptions, refer to the Communications Management  book.

When the QOLELINK API returns, your application program should examine the codes to determine the status of the link. Successful return and reason codes (both zero) indicate the link is being enabled and an enable-complete entry will be sent to the data queue or user queue specified on the call to the QOLELINK API when the enable operation completes. See “Enable-Complete Entry” on page 286 for more information on the enable-complete entry. Unsuccessful return and reason codes indicate the link could not be enabled and the enable-complete entry will not be sent to the data queue or user queue. “Return and Reason Codes” on page 8 provides more information on the QOLELINK API return and reason codes.

## Authorities and Locks

*User Space Authority*

\*READ

*User Space Library Authority*

\*USE and \*ADD. \*OBJOPR plus \*READ is equivalent to \*USE.

*User Space Lock*

\*EXCL

## Required Parameter Group

**Return code**

OUTPUT; BINARY(4)

The recovery action to take. See “Return and Reason Codes” on page 8.

**Reason code**

OUTPUT; BINARY(4)

The error that occurred. See “Return and Reason Codes” on page 8.

#### **Data unit size**

OUTPUT; BINARY(4)

The total number of bytes allocated for each data unit in the input and output buffers. For token-ring links, this includes user data (LAN user data size parameter), general LAN header information, and optional routing information. For Ethernet, wireless, and FDDI links, this includes user data (LAN user data size parameter) and general LAN header information. For X.25 links, this includes user data (X.25 user data size parameter). For more information on the general LAN header, see Return and Reason Codes for the QOLELINK API (page 25).

#### **Data units created**

OUTPUT; BINARY(4)

The number of data units created for the input buffer and the output buffer. This parameter also specifies the number of elements created for the input buffer descriptor and the output buffer descriptor. The only valid value is:

8

All protocols

**Note:** Because user-defined communications support always returns an 8, you should write your application program to avoid having to recompile should this value ever change.

#### **LAN user data size**

OUTPUT; BINARY(4)

The number of bytes allocated for token ring, Ethernet, wireless, or FDDI in each data unit of the input and output buffers. This does not include general LAN header information and optional routing information.

The content of this parameter is only valid when enabling a token-ring, Ethernet, wireless, or FDDI link.

**Note:** The maximum amount of token-ring, Ethernet, wireless, or FDDI user data that can be sent or received in each data unit is determined on a service access point basis in the line description or by the 1502 byte maximum for Ethernet Version 2 frames, and may be less than the LAN user data size. See “Query Line Description (QOLQLIND) API” on page 10 for information on retrieving these values.

#### **X.25 data unit size**

INPUT; BINARY(4)

The number of bytes allocated for X.25 user data in each data unit of the input and output buffers. This is equal to the maximum amount of X.25 user data that can be sent or received in each data unit. The content of this parameter is only valid when enabling an X.25 link.

*Range*

512 bytes-4096 bytes

#### **Input buffer**

INPUT; CHAR(20)

The name and library of the input buffer that the QOLELINK API creates for this link. The first 10 characters specify the name for the input buffer and the second 10 characters specify the name of an existing library that the input buffer will be created in. Both entries are left-justified. The special values of \*LIBL and \*CURLIB can be used for the library name.

**Note:** A user space object with the same name as the input buffer must not already exist in the specified library.

#### **Input buffer descriptor**

INPUT; CHAR(20)

The name and library of the input buffer descriptor that the QOLELINK API creates for this link. The first 10 characters specify the name of the input buffer descriptor and the second 10 characters specify the name of an existing library that the input buffer descriptor will be created in. Both entries are left-justified. The special values of \*LIBL and \*CURLIB can be used for the library name.

**Note:** A user space object with the same name as the input buffer descriptor must not already exist in the specified library.

### **Output buffer**

INPUT; CHAR(20)

The name and library of the output buffer that the QOLELINK API creates for this link. The first 10 characters specify the name of the output buffer and the second 10 characters specify the name of an existing library that the output buffer will be created in. Both entries are left-justified. The special values of \*LIBL and \*CURLIB can be used for the library name.

**Note:** A user space object with the same name as the output buffer must not already exist in the specified library.

### **Output buffer descriptor**

INPUT; CHAR(20)

The name and library of the output buffer descriptor that the QOLELINK API creates for this link. The first 10 characters specify the name of the output buffer descriptor and the second 10 characters specify the name of an existing library that the output buffer descriptor will be created in. Both entries are left-justified. The special values of \*LIBL and \*CURLIB can be used for the library name.

**Note:** A user space object with the same name as the output buffer descriptor must not already exist in the specified library.

### **Key length**

INPUT; BINARY(4)

The key length when using a keyed data queue or user queue.

0	The data queue or user queue is not keyed.
<i>Range</i>	1-256

### **Key value**

INPUT; CHAR(256)

The key value (left justified) when using a keyed data queue or user queue.

### **Qualified queue name**

INPUT; CHAR(20)

The name and library of the data queue or user queue where the enable-complete, disable-complete, permanent-link-failure, and incoming-data entries for this link will be sent. See "Queue Entries" on page 285 for more information about these queue entries. The first 10 characters specify the name of an existing queue and the second 10 characters specify the library in which the queue is located. Both entries are left-justified. The special values of \*LIBL and \*CURLIB can be used for the library name.

### **Line description**

INPUT; CHAR(10)

The name of the line description that describes the communications line the link being enabled will use. An existing token-ring, Ethernet, wireless, FDDI, or X.25 line description must be used.

### **Communications handle**

INPUT; CHAR(10)

The name assigned to the link being enabled. Any name complying with system object naming conventions may be used.

## Optional Parameter Group

### Queue type

INPUT; CHAR(1)

The type of queue you specified for the Queue name parameter.

*D* Data queue  
*U* User queue

### Network interface description

INPUT; CHAR(10)

The name of the network interface description. This value is specified if you are running X.25 and need to specify a particular network interface to use. Otherwise, this value should be set to blanks.

**Note:** This parameter along with the line description parameter causes only the network interface description specified to be varied on. If this value is not specified and the line description parameter contains a switched network interface list, all network interface descriptions within the list are varied on when the QOLELINK API is called.

Specifying this parameter causes only the line and the network interface that are passed to be varied on during enable processing.

### Extended operations

INPUT; CHAR(1)

Indicates whether or not extended operations are supported.

Extended operations affect all connections (UCEPs, PCEPs) on the link. X'B311' and X'B111' are receive extended operations. X'B110' is a send extended operation.

*1* Operations supported  
*0* Operations not supported

## Return and Reason Codes

### Return and Reason Codes for the QOLELINK API

Return / Reason Code	Meaning	Recovery
0/0	Operation successful, link enabling.	Wait to receive the enable-complete entry from the data queue or user queue before doing input/output on this link.
81/9999	Internal system error detected. Escape message CPF91F0 will be sent to the application program when this return and reason code is received.	See messages in the job log for further information. Then, report the problem using the ANZPRB command.
82/1000	User data size not valid for X.25 link.	Correct the X.25 user data size parameter. Then, try the request again.
82/1001	Key length not valid.	Correct the key length parameter. Then, try the request again.
82/1002	Queue name not valid.	Correct the queue name parameter. Then, try the request again.
82/1003	Communications handle not valid.	Correct the communications handle parameter. Then, try the request again.



Return / Reason Code	Meaning	Recovery
82/1012	Queue type not valid.	Queue type must be D or U. Correct the queue type and try the request again.
82/1013	Extended operations value not valid.	Extended operations value must be 1 or 0. Correct the extended operations value and try the request again.
82/1020	Group parameters not valid (not all the parameters within a group were passed).	Pass all parameters within the group and try the operation again.
82/2000	Line name not valid or protocol is not supported.	The line name specified must be for a line of type Ethernet, wireless, token ring, FDDI, or X.25. Correct the line name and try the request again.
82/2001	Line description, network controller description, or network device description not in a valid state.	See messages in the job log indicating the affected object and recommended recovery. Do the recovery, and try the request again.
82/2002	Not authorized to the line description or network controller description.	See messages in the job log indicating the affected object and get authorization to it. Then, try the request again.
82/2003	Could not allocate the network device description.	Try the request again. If the problem continues, report the problem using the ANZPRB command.
82/2004	Could not create the network controller description or network device description.	See messages in the job log indicating the affected object and recommended recovery. Do the recovery, and try the request again.
82/2005	Could not vary on the network interface, line description, network controller description, or network device description.	See messages in the job log indicating the affected object and recommended recovery. Do the recovery, and try the request again.
82/2006	Line description not found.	Correct the line description parameter. Then, try the request again.
82/2007	Line description damaged.	Delete and re-create the line description. Then, try the request again.
82/2008	Unsupported interface. An error occurred that indicated the network interface specified cannot be associated with the line specified. For example, you specified a network interface for a token-ring, Ethernet, or wireless line.	The network interface value is not correct for the line name value. Correct the configuration or your application.
82/2009	Network interface description not found.	Specify the correct network interface name and try the request again.
82/2010	Network interface description specified could not be used.	Check the network interface description for possible errors. Correct any errors and try the request again.
82/2400	An error occurred while creating the input buffer, input buffer descriptor, output buffer, or output buffer descriptor.	See messages in the job log indicating the affected object and recommended recovery. Do the recovery, and try the request again.
82/3000	Communications handle already assigned to another link that is enabled in this job.	Either disable the link that was assigned this communications handle, or correct the communications handle parameter so it does not specify a communications handle that is already assigned to a link enabled in this job. Then, try the request again.
82/3005	Line description already in use by another link that is enabled in this job.	Disable the link that is using this line description. Then, try the request again.

## Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF91F0 E	Internal system error.

**Message ID    Error Message Text**

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

API introduced: V2R1

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

---

## Query Line Description (QOLQLIND) API

Required Parameter Group:	
1	Return code
<b>Output</b>	Binary(4)
2	Reason code
<b>Output</b>	Binary(4)
3	Number of bytes
<b>Output</b>	Binary(4)
4	User buffer
<b>Output</b>	Char(*)
5	Line description
<b>Input</b>	Char(10)
6	Format
<b>Input</b>	Char(1)
Optional Parameter Group:	
7	Length of user buffer
<b>Input</b>	Binary(4)
8	Bytes available
<b>Output</b>	Binary(4)
Default Public Authority: *USE	
Threadsafe: No	

The Query Line Description (QOLQLIND) API queries an existing token-ring, Ethernet, wireless, FDDI, frame relay, or X.25 line description. The data received from the query is placed in the user buffer parameter.

The line description to be queried does not have to be associated with any links the application program has enabled. However, data in the line description may change after it is queried.

### Required Parameter Group

**Return code**

OUTPUT; BINARY(4)

The recovery action to take. See “Return and Reason Codes” on page 20.

**Reason code**

OUTPUT; BINARY(4)

The error that occurred. See “Return and Reason Codes” on page 20.

**Number of bytes**

OUTPUT; BINARY(4)

The number of bytes of data returned in the user buffer.

**User buffer**

OUTPUT; CHAR(\*)

The buffer where the data from the query will be received. Any unused space in the buffer will be filled with X'00'. The length of this character structure is determined using User Buffer Format (page 11).

**User Buffer Format**

Format	Group Parameter Passed	Length of Char(*)
1	No	256
1 or 2	Yes	Specified by the length user buffer parameter.

**Note:** You are recommended to set the length user buffer value to a number large enough to hold the system maximum values of virtual circuits, SAPs, and group addresses with additional space left for future needs.

**Line description**

INPUT; CHAR(10)

The name of the line description to query. An existing token-ring, Ethernet, wireless, FDDI, frame relay, or X.25 line description must be used.

**Format**

INPUT; CHAR(1)

The format of the data returned in the user buffer. The valid values are as follows:

X'01'                      Use format 01.  
X'02'                      Use format 02.

See “Format of Data in the User Buffer” on page 12 for more information.

## Optional Parameter Group

**Length of user buffer**

INPUT; BINARY(4)

The number of bytes available for the API to use in the user buffer parameter. The valid values are from 0 to 32,767.

**Notes:**

1. This parameter is required if format 2 is specified in the format parameter. It is optional if format 1 is specified.
2. If length user buffer is specified, bytes available must also be specified.
3. If additional information exists that could not be reported, the bytes available parameter will contain a larger value than the bytes returned parameter.

**Bytes available**

OUTPUT; BINARY(4)

The total number of bytes of available information.

**Notes:**

1. This parameter is required if format 2 is specified in the format parameter. It is optional if format 1 is specified.
2. If bytes available is specified, length user buffer must also be specified.
3. If the bytes available parameter contains a number larger than the bytes returned parameter, there is additional information that the application cannot access.
4. If the return code parameter is nonzero, this value is set to zero.

## Format of Data in the User Buffer

The data received in the user buffer from the query is made up of two parts. The first portion starts at offset 0 from the top of the user buffer and contains general query data. The format of this data does not depend on value of the format parameter supplied to the QOLQLIND API.

### General Query Data

Field	Type	Description
Line description	CHAR(10)	The name of the token-ring, Ethernet, wireless, FDDI, frame relay, or X.25 line description that was queried.
Line type	CHAR(1)	The type of line description that was queried. The valid values are as follows:  <i>X'04'</i> X.25 <i>X'05'</i> Token-ring <i>X'09'</i> Ethernet <i>X'0D'</i> FDDI <i>X'0E'</i> Frame relay <i>X'10'</i> Wireless
Status	CHAR(1)	The current status of the line description. The valid values are as follows:  <i>X'00'</i> Varied off <i>X'01'</i> Varied off pending <i>X'02'</i> Varied on pending <i>X'03'</i> Varied on <i>X'04'</i> Active <i>X'05'</i> Connect pending <i>X'06'</i> Recovery pending <i>X'07'</i> Recovery canceled <i>X'08'</i> Failed <i>X'09'</i> Diagnostic mode <i>X'FF'</i> Unknown

The second portion of the user buffer starts immediately after the general query data and contains data specific to the type of line description that was queried. The format of this data depends on the value of the format parameter supplied to the QOLQLIND API.

### LAN Specific Data-Format 01

Field	Type	Description
Local adapter address	CHAR(6)	Specifies, in packed form, the local adapter address of this line. The special value of X'000000000000' indicates that the preset default address for the adapter card was configured. However, the line description must be varied on before this address can be retrieved.
Line speed	CHAR(1)	The speed of this line. The valid values are as follows:  X'01' 4 megabits/second X'02' 10 megabits/second X'03' 16 megabits/second X'04' 100 megabits/second
Line capability	CHAR(1)	The capability of this line. The valid values are as follows:  X'00' Non-Ethernet X'01' Ethernet Version 2 X'02' Ethernet 802.3 X'03' Both Ethernet Version 2 and Ethernet 802.3
Line frame size	BINARY(2)	The maximum frame size possible on this line.
Ethernet Version 2 frame size	BINARY(2)	The maximum size for Ethernet Version 2 frames. This will be 1502 if the line is capable of Ethernet Version 2 traffic. Otherwise, it will be zero.
Number of SSAPs	BINARY(2)	The number of source service access points (SSAPs) configured for this line.
<b>Note:</b> The following 3 rows are repeated for each SSAP configured for this line.		
SSAP	CHAR(1)	The configured source service access point.
SSAP type	CHAR(1)	The SSAP type. The valid values are as follows:  X'00' Non-SNA SSAP X'01' SNA SSAP
SSAP frame size	BINARY(2)	The maximum frame size allowed on this SSAP.
Number of group addresses	BINARY(2)	The number of group addresses configured for this line.  <b>Note:</b> This will always be zero for a token-ring line description.
<b>Note:</b> The following row is repeated for each group address configured for this line.		
Group address	CHAR(6)	Specifies a group address, in packed form.

### LAN Specific Data-Format 02

Field	Type	Description
Local adapter address	CHAR(6)	Specifies, in packed form, the local adapter address of this line. The special value of X'000000000000' indicates that the preset default address for the adapter card was configured. However, the line description must be varied on before this address can be retrieved.

Field	Type	Description
Line speed	CHAR(1)	The speed of this line. The valid values are as follows:  X'01' 4 megabits/second X'02' 10 megabits/second X'03' 16 megabits/second X'04' 100 megabits/second X'05' Frame relay (line speed is specified separately)
Line capability	CHAR(1)	The capability of this line. The valid values are as follows:  X'00' Non-Ethernet X'01' Ethernet Version 2 X'02' Ethernet 802.3 X'03' Both Ethernet Version 2 and Ethernet 802.3
Line frame size	BINARY(2)	The maximum frame size possible on this line.
Ethernet Version 2 frame size	BINARY(2)	The maximum size for Ethernet Version 2 frames. This will be 1502 if the line is capable of Ethernet Version 2 traffic. Otherwise, it will be zero.
Functional address field	CHAR(6)	The hexadecimal functional address configured for the line. An address of X'000000000000' indicates there are no functional addresses configured on this line description.
<b>Note:</b> For additional information on functional addresses, refer to the <i>Token-Ring Architecture Reference</i> book, SC30-3374.		
Number of group addresses	BINARY(2)	The number of group addresses configured for this line. This value is valid for Ethernet and wireless line descriptions only.
Offset to group addresses	BINARY(2)	Offset within this structure to the array of group addresses
Number of SSAPs	BINARY(2)	The number of SSAPs configured for this line.
Offset to SSAPs	BINARY(2)	Offset within this structure to the array of SSAPs
FR line speed	BINARY(4)	Frame relay line speed. This value is valid only when the line type field is set to X'0E'.
Reserved	CHAR(*)	Reserved for extension
<b>Note:</b> The following row is duplicated by the number of group addresses.		
Group address	CHAR(6)	Specifies a group address, in packed form.
<b>Note:</b> The following three rows are duplicated by the number of SSAPs.		
SSAP	CHAR(1)	The configured source service access point.
SSAP type	CHAR(1)	The SSAP type. The valid values are as follows:  X'00' Non-SNA SSAP X'01' SNA SSAP
SSAP frame size	BINARY(2)	The maximum frame size allowed on this SSAP.

## X.25 Specific Data-Format 01

Field	Type	Description
Local network address length	CHAR(1)	Specifies, in hexadecimal, the number of binary coded decimal (BCD) digits in the local network address.

Field	Type	Description
Local network address	CHAR(9)	Specifies, in BCD, the local network address of this line.
Extended network addressing	CHAR(1)	Specifies whether network addressing is extended to permit the use of 17 digits in an address. The valid values are as follows:  X'01' Network addresses may be up to 15 digits X'02' Network addresses may be up to 17 digits
Address insertion	CHAR(1)	Specifies whether the system inserts the local network address in call request and call accept packets. The valid values are as follows:  'Y' The local network address is inserted in call request and call accept packets. 'N' The local network address is not inserted in call request and call accept packets.
Modulus	CHAR(1)	The X.25 modulus value. The valid values are as follows:  X'01' Modulus 8 X'02' Modulus 128
X.25 DCE support	CHAR(1)	Specifies whether the system communicates using the integrated X.25 DCE support. This allows the system, acting as the DCE, to communicate with another system without going through an X.25 network. The valid values are as follows:  X'01' The system does not communicate using the X.25 DCE support X'02' The system does communicate using the X.25 DCE support X'03' The system negotiates whether it communicates using the X.25 DCE support.
Transmit maximum packet size	BINARY(2)	The transmit maximum packet size configured for this line.
Receive maximum packet size	BINARY(2)	The receive maximum packet size configured for this line.
Transmit default packet size	BINARY(2)	The transmit default packet size configured for this line.
Receive default packet size	BINARY(2)	The receive default packet size configured for this line.
Transmit default window size	BINARY(1)	The transmit default window size configured for this line.
Receive default window size	BINARY(1)	The receive default window size configured for this line.
Number of logical channels	BINARY(2)	The number of logical channels configured for this line.
<b>Note:</b> The following 4 rows are repeated for each logical channel configured for this line		
Logical channel group number	CHAR(1)	The logical channel group number. This together with the logical channel number makes up the logical channel identifier.
Logical channel number	CHAR(1)	The logical channel number. This together with the logical channel group number makes up the logical channel identifier.

Field	Type	Description
Logical channel type	CHAR(1)	<p>The logical channel type. The valid values are as follows:</p> <p><i>X'01'</i> Switched virtual circuit (SVC).</p> <p><i>X'02'</i> Permanent virtual circuit (PVC) that is eligible for use by a network controller.</p> <p><b>Note:</b> This does not necessarily mean that this PVC is available for use. Another job running on the network controller attached to this line may already have this PVC in use.</p> <p><i>X'22'</i> PVC that is not eligible for use by a network controller. For example, a PVC that is already attached to an asynchronous controller description.</p>
Logical channel direction	CHAR(1)	<p>The direction of calls allowed on the logical channel. The valid values are as follows:</p> <p><i>X'00'</i> Not applicable (PVC logical channel).</p> <p><i>X'01'</i> Only incoming calls are allowed on this logical channel.</p> <p><i>X'02'</i> Only outgoing calls are allowed on this logical channel.</p> <p><i>X'03'</i> Both incoming and outgoing calls are allowed on this logical channel.</p>

## X.25 Specific Data-Format 02

Field	Type	Description
Local network address length	CHAR(1)	Specifies, in hexadecimal, the number of binary coded decimal (BCD) digits in the local network address.
Local network address	CHAR(9)	Specifies, in BCD, the local network address of this line.
Extended network addressing	CHAR(1)	<p>Specifies whether network addressing is extended to permit the use of 17 digits in an address. The valid values are as follows:</p> <p><i>X'01'</i> Network addresses may be up to 15 digits</p> <p><i>X'02'</i> Network addresses may be up to 17 digits</p>
Address insertion	CHAR(1)	<p>Specifies whether the system inserts the local network address in call request and call accept packets. The valid values are as follows:</p> <p><i>'Y'</i> The local network address is inserted in call request and call accept packets.</p> <p><i>'N'</i> The local network address is not inserted in call request and call accept packets.</p>



Field	Type	Description
Modulus	CHAR(1)	The X.25 modulus value. The valid values are as follows:  X'01'    Modulus 8 X'02'    Modulus 128
X.25 DCE support	CHAR(1)	Specifies whether the system communicates using the integrated X.25 DCE support. This allows the system, acting as a DCE, to communicate with another system without going through an X.25 network. The valid values are as follows:  X'01'    The system does not communicate using the X.25 DCE support X'02'    The system does communicate using the X.25 DCE support X'03'    The system negotiates whether it communicates using the X.25 DCE support.
Transmit maximum packet size	BINARY(2)	The transmit maximum packet size configured for this line.
Receive maximum packet size	BINARY(2)	The receive maximum packet size configured for this line.
Transmit default packet size	BINARY(2)	The transmit default packet size configured for this line.
Receive default packet size	BINARY(2)	The receive default packet size configured for this line.
Transmit default window size	BINARY(1)	The transmit default window size configured for this line.
Receive default window size	BINARY(1)	The receive default window size configured for this line.
Number of logical channels	BINARY(2)	The number of logical channels configured for this line.
Maximum frame size	BINARY(2)	The maximum frame size configured in the line description. The valid values are as follows: • 1024 • 2048 • 4096
ISDN interface	CHAR(1)	Indicates if the line uses an ISDN interface. The valid values are as follows:  X'00'    X.25 line does not run over an ISDN interface. X'01'    X.25 line runs over an ISDN interface.
<b>Note:</b> The following section applies only if the ISDN interface is specified as X'01'. The sections of format 02 on the call direction field to the offset to logical channel array field are not meaningful if an ISDN interface is not used and will return zeros in these fields if an ISDN interface is not specified.		
Call direction	CHAR(1)	The direction of the ISDN call. The valid values are as follows:  X'00'    Incoming switched call X'01'    Outgoing switched call X'02'    Either a nonswitched call or not ISDN-capable.
<b>Note:</b> The following fields are only meaningful if the line description is switched.		
Length of call ID information	BINARY(2)	Length includes type and plan, as described below, and the call identify information element.

Field	Type	Description
Type of number and numbering plan	BINARY(1)	<p>Type and plan as represented by the following bit sequence: tttt pppp, where tttt equals the category of the calling number and pppp equals the numbering plan identification used when the calling party number was created.</p> <p><i>Type '0000 xxxx'</i> Unknown number</p> <p><i>Type '0001 xxxx'</i> International number</p> <p><i>Type '0010 xxxx'</i> National number</p> <p><i>Type '0011 xxxx'</i> Network specific number</p> <p><i>Type '0100 xxxx'</i> Subscriber number</p> <p><i>Type '0110 xxxx'</i> Abbreviated number</p> <p><i>Type '0111 xxxx'</i> Reserved for extension</p> <p><i>Plan 'xxxx 0000'</i> Unknown</p> <p><i>Plan 'xxxx 0001'</i> ISDN/telephony numbering plan</p> <p><i>Plan 'xxxx 0011'</i> Data numbering plan</p> <p><i>Plan 'xxxx 0100'</i> Telex numbering plan</p> <p><i>Plan 'xxxx 1000'</i> National standard numbering plan</p> <p><i>Plan 'xxxx 1001'</i> Private numbering plan</p> <p><i>Plan 'xxxx 1111'</i> Reserved for extension</p> <p><b>Note:</b> Refer to CCITT Recommendation Q.931 for more information.</p>
Reserved	BINARY(1)	Reserved for extension.
Call ID digits	CHAR(128)	Calling party number of remote system received off the D-channel, specified in IA5 code (ASCII).
Length of subaddress information	BINARY(2)	Length includes type, odd-even indicator, and the subaddress information element. Values can range from X'0001' to X'00FF'. The user specified subaddress is restricted to 20 bytes.

Field	Type	Description
Type of subaddress and odd-even indicator	BINARY(1)	Type and odd-even indicator as represented by the following bit sequence: tttt ixxx, where tttt equals the type of subaddress and i equals whether the address has an even or odd number of digits.  <i>Type '0000 xxxx'</i> NSAP <i>Type '0010 xxxx'</i> User specified <i>Type remaining</i> Reserved <i>Plan 'xxxx 0xxx'</i> Even number of address digits <i>Plan 'xxxx 1xxx'</i> Odd number of address digits  <b>Note:</b> Refer to CCITT Recommendation Q.931 for more information.
Reserved	BINARY(1)	Reserved for extension.
Subaddress	CHAR(128)	Calling party subaddress information, received from the D-channel, specified in the IA5 code set (a superset of ASCII).
Offset to logical channel array	BINARY(2)	Offset within this structure to the array of logical channels
Reserved	CHAR(*)	Reserved for extension
<b>Note:</b> The following 5 rows are repeated for each logical channel configured for this line. This section is not specific to ISDN interfaces.		
Logical channel group number	CHAR(1)	The logical channel group number. This together with the logical channel number makes up the logical channel identifier.
Logical channel number	CHAR(1)	The logical channel number. This together with the logical channel group number makes up the logical channel identifier.
Logical channel type	CHAR(1)	The logical channel type. The valid values are as follows:  <i>X'01'</i> Switched virtual circuit (SVC). <i>X'02'</i> Permanent virtual circuit (PVC) that is eligible for use by a network controller. <b>Note:</b> This does not necessarily mean that this PVC is available for use. Another job running on the network controller attached to this line may already have this PVC in use.
Type of calls allowed	CHAR(1)	Types of calls supported on the logical channel. The valid values are as follows:  <i>X'00'</i> Not applicable (PVC logical channel). <i>X'01'</i> Only incoming calls are allowed on this logical channel. <i>X'02'</i> Only outgoing calls are allowed on this logical channel. <i>X'03'</i> Both incoming and outgoing calls are allowed on this logical channel.

Field	Type	Description
Availability	CHAR(1)	Specifies whether the virtual circuit is available or currently is in use. The valid values are as follows:  X'00' Available X'01' In use

## Return and Reason Codes

### Return and Reason Codes for the QOLQLIND API

Return / Reason Code	Meaning	Recovery
00/0000	Operation successful.	Continue processing.  <b>Notes:</b> 1. When calling QOLQLIND (specifying an X.25 line description, format 1, and not specifying group parameters), up to 54 logical channels can be contained in the user buffer because it is limited to a size of 256 bytes. To increase the size of the user buffer so that it is sufficient to contain all of the logical channels, the group parameters should be used. To determine if there are more than 54 logical channels configured, use the Display Line Description (DSPLIND) command. 2. The application should check to ensure that the bytes available value returned is less than or equal to the bytes returned value. If so, there is additional information that the application may want to receive. To receive this information, the application must re-issue the call, specifying the length user buffer equal to or greater than the bytes available value.
81/9999	Internal system error detected. Escape message CPF91F0 will be sent to the application program when this return and reason code is received.	See messages in the job log for further information. Report the problem using the ANZPRB command.
83/1005	Format not valid.	Correct the format parameter. Try the request again.
83/1014	Length user buffer value not valid. This value cannot be negative.	Correct the length user buffer value to a zero or a positive value less than 32K and try the operation again.
83/1020	Group parameters not valid.	All parameters within the group must be specified. Correct the parameter list and try the request again.
83/1021	Required parameter not specified.	Format 2 was requested and the required group parameters (length user buffer and bytes available) were not specified. Correct the parameter list and try the request again.
83/1998	User buffer parameter too small.	Either the length user buffer value is negative or it contains a positive value and the system was not able to put the data into the user buffer provided by the application. Correct the application and try the request again.
83/2000	Line description not configured for token-ring, Ethernet, wireless, or X.25.	Correct the line description parameter. Try the request again.
83/2002	Not authorized to line description.	Get authorization to the line description. Try the request again.
83/2006	Line description not found.	Correct the line description parameter. Try the request again.
83/2007	Line description damaged.	Delete and re-create the line description. Try the request again.

## Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF91F0 E	Internal system error.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

[Top](#) | [“Communications APIs,”](#) on page 1 | [APIs by category](#)

---

## Receive Data (QOLRECV) API

Required Parameter Group:	
1	Return code
<b>Output</b>	Binary(4)
2	Reason code
<b>Output</b>	Binary(4)
3	Existing user connection end point ID
<b>Output</b>	Binary(4)
4	New provider connection end point ID
<b>Output</b>	Binary(4)
5	Operation
<b>Output</b>	Char(2)
6	Number of data units
<b>Output</b>	Binary(4)
7	Data available
<b>Output</b>	Char(1)
8	Diagnostic data
<b>Output</b>	Char(40)
9	Communications handle
<b>Input</b>	Char(10)
	Default Public Authority: *USE
Threadsafe: No	

The Receive Data (QOLRECV) API performs an input operation on a link that is currently enabled in the job in which the application program is running. The type of data received is returned in the operation parameter. The data itself, is returned in the input buffer that was created when the link was enabled. For X'0001' operations, a description of that data is also be returned in the input buffer descriptor that is created when the link was enabled.

The QOLRECV API can receive different types of data depending on the type of communications line the link is using. See “LAN Input Operations” on page 25 for more information on the types of data that can

be received on links using a token-ring, Ethernet, wireless, or FDDI communications line. See “X.25 SVC and PVC Input Operations” on page 27 for more information on the types of data that can be received on links using an X.25 communications line.

**Note:** The QOLRECV API should only be called when the user-defined communications support has data available to be received. This is indicated either by an incoming-data entry on the data queue or user queue, or by the data available parameter on the QOLRECV API.

## Required Parameter Group

### Return code

OUTPUT; BINARY(4)

The recovery action to take. See “Return and Reason Codes” on page 34.

### Reason code

OUTPUT; BINARY(4)

The error that occurred. See “Return and Reason Codes” on page 34.

### Existing user connection end point ID

OUTPUT; BINARY(4)

The user connection end point (UCEP) ID that the data was received on. For links using a token-ring, Ethernet, wireless, or FDDI communications line, the content of this parameter will always be 1.

For links using an X.25 communications line, the content of this parameter is only valid when the operation parameter is X'0001', X'B001', X'B101', X'B301', or X'BF01'. It will contain the UCEP ID that was provided in the new user connection end point ID parameter on the call to the QOLSEND API with operation X'B000' or X'B400'.

**Note:** If an incoming X.25 SVC call is rejected by the user-defined communications application program by calling the QOLSEND API with operation X'B100', the content of this parameter will be set to zero when notification of the completion of the X'B100' operation is received from the QOLRECV API (operation X'B101').

### New provider connection end point ID

OUTPUT; BINARY(4)

The provider connection end point (PCEP) ID for the connection that is to be established. This identifier must be used on all subsequent calls to the QOLSEND API for this connection.

The content of this parameter is only valid for links using an X.25 communications line and when the operation parameter is X'B201'.

### Operation

OUTPUT; CHAR(2)

The type of data received by the application program. With the exception of X'0001', all values are only valid for links using an X.25 communications line. The valid values are as follows:

X'0001'	User data.
X'B001'	Completion of the X'B000' output operation.
X'B101'	Completion of the X'B100' output operation.
X'B111'	Completion of the X'B110' output operation.
	Cleanup of all connections complete. No data is associated with this operation.
X'B201'	Incoming X.25 switched virtual circuit (SVC) call.
X'B301'	Connection failure or reset indication received.

X'B311'

Connection failure applying to all connections for this link.

X'BF01'

This operation is only received when the extended operations parameter for the QOLELINK API is set to operations supported. Completion of the reset (X'BF00') output operation.

**Note:** The special value of X'0000' will be returned in the operation parameter to indicate no data was received from the QOLRECV API. See "Return and Reason Codes" on page 34 for more information.

**Number of data units**

OUTPUT; BINARY(4)

The number of data units in the input buffer that contain data. Any value between 1 and the number of data units created in the input buffer may be returned when the operation parameter is X'0001'. Otherwise, any value between 0 and 1 may be returned.

**Note:** The number of data units created in the input buffer was returned in the data units created parameter on the call to the QOLELINK API. See "Enable Link (QOLELINK) API" on page 4 for more information.

**Data available**

OUTPUT; CHAR(1)

Specifies whether more data is available for the user-defined communications application program to receive. The valid values are as follows:

X'00'

No more data is available for the user-defined communications application program to receive.

X'01'

More data is available for the user-defined communications application program to receive. The QOLRECV API must be called again prior to any other operations.

**Note:** An incoming-data entry will be sent to the data queue or user queue only when the content of this parameter is X'00' and then more data is subsequently available to be received. See for more information.

**Diagnostic data**

OUTPUT; CHAR(40)

Specifies additional diagnostic data. See "Format of Diagnostic Data Parameter" for more information.

The content of this parameter is only valid when the operation parameter is X'B001', X'B101', X'B301', X'B311', or X'BF01'.

**Communications handle**


INPUT; CHAR(10)

The name of the link on which to receive the data.


**Format of Diagnostic Data Parameter**

The format of the diagnostic data parameter is shown below. The contents of the fields within this parameter are only valid on X'B001', X'B101', X'B301', X'B311', and X'BF01' operations for the indicated return and reason codes.

Field	Type	Description
Reserved	CHAR(2)	Reserved for extension.

Field	Type	Description
Error code	CHAR(4)	Specifies hexadecimal diagnostic information that can be used to determine recovery actions.  The content of this field is only valid for 83/4001 and 83/4002 return/reason codes.
Time stamp	CHAR(8)	The time the error occurred.  The content of this field is only valid for 83/4001 and 83/4002 return/reason codes.
Error log identifier	CHAR(4)	The hexadecimal identifier that can be used for locating error information in the error log.  The content of this field is only valid for 83/4001 and 83/4002 return/reason codes.
Reserved	CHAR(10)	Reserved for extension.
Indicators	CHAR(1)	The indicators that the user-defined communications application program can use to diagnose a potential error condition. This is a bit-sensitive field.  The valid values for bit 0 (leftmost bit) are as follows:  '0'B     Either there is no message in the QSYSOPR message queue, or there is a message and it does not have the capability to run problem analysis report (PAR) to determine the cause of the error.  '1'B     There is a message in the QSYSOPR message queue for this error, and it does have the capability to run problem analysis report (PAR) to determine the cause of the error.  The valid values for bit 1 are as follows:  '0'B     The line error can be retried.  '1'B     The line error is not able to be restarted.  The valid values for bit 2 are as follows:  '0'B     The cause and diagnostic codes fields are not valid.  '1'B     The cause and diagnostic codes fields are valid.  The valid values for bit 3 are as follows:  '0'B     The error has not been reported to the system operator message queue.  '1'B     The error has been reported to the system operator message queue.  The valid values for bit 4 are as follows:  '0'B     A reset request packet was transmitted on the network  '1'B     A reset confirmation packet was transmitted on the network instead of a reset request packet.  The content of bit 4 is only valid for operation X'BF01' with 00/0000 return/reason codes.  The content of the indicators field is only valid for 83/4001, 83/4002, and 83/3202 return/reason codes, and 00/0000 return/reason codes for operation X'BF01'.
X.25 cause code	CHAR(1)	Specifies additional information on the condition reported. See the X.25 Network Support  book for interpreting the values of this field.  The content of this field is only valid for 83/4001, 83/4002 and 83/3202 return/reason codes.



Field	Type	Description
X.25 diagnostic code	CHAR(1)	Specifies additional information on the condition reported. See the X.25 Network Support  book for interpreting the values of this field. The content of this field is only valid for 83/4001, 83/4002 and 83/3202 return/reason codes.
Reserved	CHAR(1)	Reserved for extension.
Error offset	BINARY(4)	The offset from the top of the input buffer to the incorrect data in the input buffer. The content of this field is only valid for a 83/1999 return/reason code.
Reserved	CHAR(4)	Reserved for extension.

## LAN Input Operations

The only type of data that an application program can receive from the QOLRECV API on links using a token-ring, Ethernet, wireless, or FDDI communications line is user data (operation X'0001'). User-defined communications support returns the following information for each data frame received from the QOLRECV API:

- One or more data units. The first data unit contains a general LAN header, routing information if a token ring is used, and user data.
- Total length of the data unit. This information is reported in the corresponding input buffer descriptor element.

For example, suppose two data frames came in from the network and the user-defined communications application program was notified of this by an incoming-data entry on the data queue or user queue. On return from the QOLRECV API, the information for the first frame would be in the first data unit of the input buffer and described in the first element of the input buffer descriptor. The information for the second frame would be in the second data unit of the input buffer and described in the second element of the input buffer descriptor. The number of data units parameter would be set to 2.

### Data Unit Format-LAN Operation X'0001'

Each data frame received from the QOLRECV API corresponds to a data unit in the input buffer. The information in each of these data units is made up of a general LAN header, routing information (for token-ring links only), followed by user data.

The general LAN header is used to pass information about the frame to the communications support. The fields in the general LAN header are used for all LAN link types, although some of them are link specific. For example, routing information is only for token-ring links, and the length of routing information is X'00' to X'18'. For non-token-ring links, the length of the routing information is always X'00'. Also, DSAP and SSAP are defined for protocols that use the 802.2 logical link control interface and do not apply to Ethernet Version 2. A DSAP and SSAP of X'00' tells the communications support that the data frame is an Ethernet Version 2 frame.

### Format of the General LAN Information

Field	Type	Description
Length of general LAN information	BINARY(2)	The length of the general LAN information in the data unit, including this field. This field is always set to 16.

Field	Type	Description
Sending adapter address	CHAR(6)	Specifies, in packed form, the adapter address from which this frame was sent. The possible values returned in this field depend on the filters activated for this link. See "Set Filter (QOLSETF) API" on page 60 for more information.  <b>Note:</b> Because user-defined communications support only allows connectionless service over LANs, all frames received on a single call to the QOLRECV API may not have the same source adapter address.
DSAP address	CHAR(1)	The service access point on which the iSeries server received this frame. The possible values returned in this field depend on the filters activated for this link. See "Set Filter (QOLSETF) API" on page 60 for more information.  <b>Note:</b> The Ethernet Version 2 standard does not define a DSAP address in an Ethernet Version 2 frame. Therefore, when receiving Ethernet Version 2 frames, the DSAP address will be null (X'00').
SSAP address	CHAR(1)	The service access point on which the source system sent this frame. The possible values returned in this field depend on the filters activated for this link. See "Set Filter (QOLSETF) API" on page 60 for more information.  <b>Note:</b> The Ethernet Version 2 standard does not define a SSAP address in an Ethernet Version 2 frame. Therefore, when receiving Ethernet Version 2 frames, the SSAP address will be null (X'00').
Reserved	CHAR(2)	Reserved for extension.
Length of token-ring routing information	BINARY(2)	The length of the routing information in the data unit. For links using a token-ring communications line, any value between 0 and 18 may be returned, where 0 indicates that there is no routing information.  For links using an Ethernet, wireless, or FDDI communications line, the content of this field is not applicable and will be set to 0 indicating that there is no routing information.
Length of user data	BINARY(2)	The length of the user data in the data unit. This will be less than or equal to the maximum frame size allowed on the service access point returned in the DSAP address field. See "Query Line Description (QOLQLIND) API" on page 10 to determine the maximum frame size allowed on the service access point returned in the DSAP address field.  For Ethernet Version 2 frames, this will be at least 48 and not more than 1502 (including 2 bytes for the Ethernet type field).  <b>Note:</b> Ethernet 802.3 frames will be padded when the user data is less than 46 bytes.

Token-ring routing information follows the general LAN header. The length of this field is specified by the length of token-ring routing information field found in the general LAN header. If the length of the routing information is nonzero, the user data follows the routing information header.

The following table shows the fields and offsets used for Ethernet 802.3, wireless, and token-ring frames without routing information.

General LAN Header	User Data
0	16

The length of the user data is described in the length of user data field in the general LAN header. For Ethernet Version 2 frames, the first 2 bytes of user data are used for the *frame type*. The type field is a 2-byte field that specifies the upper layer protocol of the frame.

The adapter address, DSAP, SSAP, and frame type fields are all used to define inbound routing information used by the QOLSETF API. Refer to “Set Filter (QOLSETF) API” on page 60 for information on the QOLSETF API and how inbound routing information is used to route inbound data to the application program.

**Note:** Inbound routing information is not related to the token-ring routing information described in the general LAN header.

The following table shows the fields and offsets used for token-ring frames with routing information.

General LAN Header	Routing Information	User Data
0	16	16 + Length of Routing Information

The following table shows the fields and offsets used for Ethernet Version 2 frames.

**Note:** For Ethernet Version 2, the frame type field is the first 2 bytes of user data, following the general LAN information, with user data starting at offset 18.

General LAN Header	User Data	
	Frame Type	Data
0	16	18

### Input Buffer Descriptor Element Format-LAN Operation X'0001'

The information returned in each data unit of the input buffer will be described in the corresponding element of the input buffer descriptor. The following table shows the format of each element in the input buffer descriptor.

Field	Type	Description
Length	BINARY(2)	The number of bytes of information in the corresponding data unit of the input buffer. This will be equal to the length of the general LAN information with the length of the routing information and the length of the user data. See Format of the General LAN Information (page 25) for general LAN information fields and descriptions.
Reserved	CHAR(30)	Reserved for extension.

## X.25 SVC and PVC Input Operations

The following table shows the types of data that can be received from the QOLRECV API on links using an X.25 communications line.

Operation	Meaning
X'0001'	User data (SVC or PVC).
X'B001'	Completion of the X'B000' output operation (SVC or PVC).
X'B101'	Completion of the X'B100' output operation (SVC or PVC).
X'B201'	Incoming X.25 call (SVC).
X'B301'	Connection failure or reset indication (SVC or PVC).
X'B311'	Connection failure applying to all connections for this link.
X'BF01'	Completion of the X'BF00' output operation (SVC or PVC).

### X.25 Operation X'0001'

This operation indicates that user data was received on an X.25 SVC or PVC connection. User-defined communications support will return the following information:

- User data in the next data unit of the input buffer, starting with the first data unit
- A description, in the corresponding element of the input buffer descriptor, of the user data in that data unit

For example, suppose two data units of user data came in from the network and the application program was notified of this by an incoming-data entry on the data queue or user queue. On return from the QOLRECV API, the first portion of the user data would be in the first data unit of the input buffer and described in the first element of the input buffer descriptor. The second portion of the user data would be in the second data unit of the input buffer and described in the second element of the input buffer descriptor. The number of data units parameter would be set to 2.

User-defined communications support will automatically reassemble the X.25 data packet(s) from a complete packet sequence into the next data unit of the input buffer. If the amount of user data in a complete packet sequence is more than what can fit into a data unit, the more data indicator field in the corresponding element of the input buffer descriptor will be set to X'01' and the next data unit will be used for the remaining user data, and so on.

#### Data Unit Format-X.25 Operation X'0001'

Each data unit in the input buffer consists solely of user data and starts offset 0 from the top of the data unit.

#### Input Buffer Descriptor Element Format-X.25 Operation X'0001'

The user data returned in each data unit of the input buffer will be described in the corresponding element of the input buffer descriptor.

Field	Type	Description
Length	BINARY(2)	<p>The number of bytes of user data in the corresponding data unit of the input buffer. This will always be less than or equal to the X.25 user data size parameter that was specified on the call to the QOLELINK API when the link was enabled. See "Enable Link (QOLELINK) API" on page 4 for more information.</p> <p><b>Note:</b> The maximum amount of user data in a data unit of the input buffer may be further limited by the maximum data unit assembly size for a connection. See "Send Data (QOLSEND) API" on page 38 for more information.</p>
More data indicator	CHAR(1)	<p>Specifies whether the remaining amount of user data from a complete X.25 packet sequence is more than can fit into the corresponding data unit. The valid values are as follows:</p> <p>X'00' The remaining amount of user data from a complete X.25 packet sequence fit into the corresponding data unit.</p> <p>X'01' The remaining amount of user data from a complete X.25 packet sequence could not all fit into the corresponding data unit. The next data unit will be used.</p>

Field	Type	Description
Qualified data indicator	CHAR(1)	Specifies whether the X.25 qualifier bit (Q-bit) was set on or off in all X.25 packets reassembled into the corresponding data unit. The valid values are as follows:  <i>X'00'</i> The Q-bit was set off in all X.25 packets reassembled into the corresponding data unit.  <i>X'01'</i> The Q-bit was set on in all X.25 packets reassembled into the corresponding data unit.
Interrupt packet indicator	CHAR(1)	Specifies whether the user data in the corresponding data unit was received in an X.25 interrupt packet. The valid values are as follows:  <i>X'00'</i> The user data in the corresponding data unit was received in one or more data packets.  <i>X'01'</i> The user data in the corresponding data unit was received in an X.25 interrupt packet.
Delivery confirmation indicator	CHAR(1)	Specifies whether the X.25 delivery confirmation bit (D-bit) was set on or off in all X.25 packets reassembled into the corresponding data unit. The valid values are as follows:  <i>X'00'</i> The D-bit was set off in all X.25 packets reassembled into the corresponding data unit.  <i>X'01'</i> The D-bit was set on in all X.25 packets reassembled into the corresponding data unit.  <b>Note:</b> A packet-level confirmation is sent by the input/output processor (IOP) when a packet is received with the X.25 D-bit set on.
Reserved	CHAR(26)	Reserved for extension.

### X.25 Operation X'B001'

This operation indicates that a X'B000' output operation has completed. User-defined communications support will return the data for this operation (if any) in the first data unit of the input buffer. The input buffer descriptor is not used.

Data will be returned in the input buffer for the following return and reason codes:

- 0/0
- 83/1999
- 83/4002 (only when the number of data units parameter is set to one)

The format of the data returned in the input buffer for the X'B001' operation depends on whether the X'B000' output operation was used to initiate an SVC call or to open a PVC connection. Each format will be explained below.

**Note:** The formats below only apply to 0/0 and 83/4002 return and reason codes. When the X'B001' operation is received with a 83/1999 return and reason code, the data returned starts at offset 0 from the top of the first data unit in the input buffer and contains the data specified in the output buffer on the X'B000' output operation. See "Send Data (QOLSEND) API" on page 38 for more information.

#### Data Unit Format-X.25 Operation X'B001' (Completion of SVC Call)

The data returned starts at offset 0 from the top of the first data unit in the input buffer.

Field	Type	Description
Reserved	CHAR(2)	Reserved for extension.
Logical channel identifier	CHAR(2)	The logical channel identifier assigned to the SVC connection. <sup>1</sup>
Transmit packet size	BINARY(2)	The negotiated transmit packet size for this connection. <sup>1</sup>
Transmit window size	BINARY(2)	The negotiated transmit window size for this connection. <sup>1</sup>
Receive packet size	BINARY(2)	The negotiated receive packet size for this connection. <sup>1</sup>
Receive window size	BINARY(2)	The negotiated receive window size for this connection. <sup>1</sup>
Reserved	CHAR(32)	Reserved for extension.
Delivery confirmation support	CHAR(1)	Specifies whether the X.25 delivery confirmation bit (D-bit) was set on or off in the call connected packet. This also specifies the D-bit support for this connection. <sup>1</sup> The valid values are as follows:  X'00' The D-bit was set off in the call connected packet. D-bit will be supported for sending data but not for receiving data.  <b>Note:</b> When this value is returned and an X.25 packet is received with the D-bit set on, the input/output processor (IOP) will send a reset packet.  X'01' The D-bit was set on in the call connected packet. D-bit will be supported for sending data and for receiving data.
Reserved	CHAR(11)	Reserved for extension.
X.25 facilities length	BINARY(1)	The number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be returned.
X.25 facilities	CHAR(109)	The X.25 facilities data.
Reserved	CHAR(48)	Reserved for extension.
Call/clear user data length	BINARY(2)	The number of bytes of data in the call/clear user data field. Any value between 0 and 128 may be returned.
Call/clear user data	CHAR(128)	For a 0/0 return and reason code, this specifies the call user data. For an 83/4002 return and reason code, this specifies the clear user data.
Reserved	CHAR(168)	Reserved for extension.
<sup>1</sup> The content of this field is only valid for a 0/0 return and reason code.		

### Data Unit Format-X.25 Operation X'B001' (Completion of Open PVC)

The data returned starts at offset 0 from the top of the first data unit in the input buffer.

Field	Type	Description
Reserved	CHAR(4)	Reserved for extension.
Transmit packet size	BINARY(2)	The negotiated transmit packet size for this connection.  <b>Note:</b> This will be the same as the requested transmit packet size specified on the X'B000' output operation.
Transmit window size	BINARY(2)	The negotiated transmit window size for this connection.  <b>Note:</b> This will be the same as the requested transmit window size specified on the X'B000' output operation.
Receive packet size	BINARY(2)	The negotiated receive packet size for this connection.  <b>Note:</b> This will be the same as the requested receive packet size specified on the X'B000' output operation.

Field	Type	Description
Receive window size	BINARY(2)	The negotiated receive window size for this connection.  <b>Note:</b> This will be the same as the requested receive window size specified on the X'B000' output operation.
Reserved	CHAR(500)	Reserved for extension.

### X.25 Operation X'B101'

This operation indicates that a X'B100' output operation has completed. User-defined communications support will return the data for this operation (if any) in the first data unit of the input buffer. The input buffer descriptor is not used.

Data will be returned in the input buffer for the following return and reason codes:

- 0/0 (only when the number of data units parameter is set to one)
- 83/1999

**Note:** The format below only applies for a 0/0 return and reason code. When the X'B101' operation is received with an 83/1999 return and reason code, the data returned starts at offset 0 from the top of the first data unit in the input buffer and contains the data specified in the output buffer on the X'B100' output operation. See "Send Data (QOLSEND) API" on page 38 for more information.

#### Data Unit Format-X.25 Operation X'B101'

The data returned starts at offset 0 from the top of the first data unit in the input buffer.

Field	Type	Description
Clear type	CHAR(2)	The type of clear user data returned. The valid values are as follows:  X'0001' Clear confirmation data included. X'0002' Clear indication data included.
Cause code	CHAR(1)	The X.25 cause code.
Diagnostic code	CHAR(1)	The X.25 diagnostic code.
Reserved	CHAR(4)	Reserved for extension.
X.25 facilities length	BINARY(1)	The number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be returned.
X.25 facilities	CHAR(109)	The X.25 facilities data.
Reserved	CHAR(48)	Reserved for extension.
Clear user data length	BINARY(2)	The number of bytes of data in the clear user data field. Any value between 0 and 128 may be returned.
Clear user data	CHAR(128)	The clear user data.
Reserved	CHAR(216)	Reserved for extension.

### X.25 Operation X'B111'

This operation indicates a X'B110' output operation has completed. All connections have been closed and the clean up of connection control information is complete. All UCEPs and PCEPs are freed. There is no data associated with this operation.

### X.25 Operation X'B201'

This operation indicates that an incoming X.25 SVC call was received. User-defined communications support returns the data for this operation in the first data unit of the input buffer. The input buffer descriptor is not used.

**Note:** It is the responsibility of the application program to either accept or reject the incoming call. This is done by calling the QOLSEND API with operation X'B400' or X'B100', respectively.

#### Data Unit Format-X.25 Operation X'B201'

The data returned starts at offset 0 from the top of the first data unit in the input buffer.

Field	Type	Description
Reserved	CHAR(2)	Reserved for extension.
Logical channel identifier	CHAR(2)	The logical channel identifier assigned to the incoming SVC call.
Transmit packet size	BINARY(2)	The requested transmit packet size for this connection.
Transmit window size	BINARY(2)	The requested transmit window size for this connection.
Receive packet size	BINARY(2)	The requested receive packet size for this connection.
Receive window size	BINARY(2)	The requested receive window size for this connection.
Reserved	CHAR(7)	Reserved for extension.
Calling DTE address length	BINARY(1)	The number of binary coded decimal (BCD) digits in the calling DTE address.
Calling DTE address	CHAR(16)	Specifies, in binary coded decimal (BCD), the calling DTE address. The address will be left justified and padded on the right with BCD zeros.
Reserved	CHAR(8)	Reserved for extension.
Delivery confirmation support	CHAR(1)	Specifies whether the X.25 delivery confirmation bit (D-bit) was set on or off in the incoming call packet. The valid values are as follows:  X'00' The D-bit was set off in the incoming call packet. X'01' The D-bit was set on in the incoming call packet.
Reserved	CHAR(9)	Reserved for extension.
Reverse charging indicator	CHAR(1)	Specifies reverse charging options. The valid values are as follows:  X'00' Reverse charging not requested. X'01' Reverse charging requested.
Fast select indicator	CHAR(1)	Specifies fast select options. The valid values are as follows:  X'00' Fast select not requested. X'01' Fast select with restriction requested. X'02' Fast select without restriction requested.
X.25 facilities length	BINARY(1)	The number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be returned.
X.25 facilities	CHAR(109)	The X.25 facilities data.
Reserved	CHAR(48)	Reserved for extension.
Call user data length	BINARY(2)	The number of bytes of data in the call user data field. Any value between 0 and 128 may be returned.



Field	Type	Description
Call user data	CHAR(128)	The call user data.  <b>Note:</b> The iSeries server treats the first byte of call user data as the protocol identifier (PID).
Called DTE address length	BINARY(1)	The number of binary coded decimal (BCD) digits in the called DTE address.
Called DTE address	CHAR(16)	Specifies, in binary coded decimal (BCD), the called DTE address. The address will be left-justified and padded on the right with BCD zeros.
Reserved	CHAR(111)	Reserved for extension.

### X.25 Operation X'B301'

This operation indicates that a failure has occurred, or a reset indication has been received, on an X.25 SVC or PVC connection. User-defined communications support will return data for this operation in the first data unit of the input buffer only on a 83/4002 return and reason code when the number of data units parameter is set to one. The input buffer descriptor is not used.

**Note:** The diagnostic data parameter will contain the X.25 cause and diagnostic codes when a reset indication is received.

#### Data Unit Format-X.25 Operation X'B301'

The data returned starts at offset 0 from the top of the first data unit in the input buffer.

Field	Type	Description
Reserved	CHAR(8)	Reserved for extension.
X.25 facilities length	BINARY(1)	The number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be returned.
X.25 facilities	CHAR(109)	The X.25 facilities data.
Reserved	CHAR(48)	Reserved for extension.
Clear user data length	BINARY(2)	The number of bytes of data in the clear user data field. Any value between 0 and 128 may be returned.
Clear user data	CHAR(128)	The clear user data.
Reserved	CHAR(216)	Reserved for extension.

### X.25 Operation X'B311'

This operation indicates that an error has occurred that has caused the system to close all connections on the link. The error may be a system error or a network error. The error information is returned in the diagnostic data and no additional data is provided.

**Note:** This operation is only received when the extended operation parameter on the QOLELINK API is set to operation supported. If the extended operations are not supported and an error occurs that will close all connections, X'B301' is received for each connection.

### X.25 Operation X'BF01'

This operation indicates that a X'BF00' output operation has been completed. Neither the input buffer nor the input buffer descriptor is used for this operation.

**Note:** When the X'BF01' operation is received with a 0/0 return and reason code, the diagnostic data parameter will contain information indicating if a reset request or reset confirmation packet was sent.

## Return and Reason Codes

The return and reason codes that can be returned from the QOLRECV API depend on the type of communications line the link is using and on the type of data (operation) that was received.

### LAN Return and Reason Codes

The following table shows the return and reason codes that indicate data could not be received from the QOLRECV API.

**Note:** When these return and reason codes are returned, all output parameters except the return and reason codes will contain hexadecimal zeros.

Return / Reason Code	Meaning	Recovery
0/3203	No data available to be received.	Ensure that user-defined communications support has data available to be received before calling the QOLRECV API. Try the request again.
80/2200	Queue error detected. Escape message CPF91F1 will be sent to the application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Correct the error, enable the link, and try the request again.
80/2401	Input buffer or input buffer descriptor error detected. Escape message CPF91F1 will be sent to the application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Correct the error, enable the link, and try the request again.
80/3002	A previous error occurred on this link that was reported to the application program by escape message CPF91F0 or CPF91F1. However, the application program has attempted another operation.	Ensure the link is disabled and see messages in the job log for further information. If escape message CPF91F0 was sent to the application program, then report the problem using the ANZPRB command. Otherwise, correct the error, enable the link, and try the request again.
80/4000	Error recovery has been canceled for this link.	Ensure the link is disabled and see messages in the job log for further information. Correct the condition, enable the link, and try the request again.
80/9999	Internal system error detected. Escape message CPF91F0 will be sent to the application program when this return and reason code is received.	See messages in the job log for further information. Report the problem using the ANZPRB command.
83/3001	Link not enabled.	Correct the communications handle parameter. Try the request again.
83/3004	Link is enabling.	Wait for the enable-complete entry to be sent to the data queue or user queue. If the link was successfully enabled, try the request again.

### Return and Reason Codes for LAN Operation X'0001'

Return / Reason Code	Meaning	Recovery
0/0	User data received successfully.	Continue processing.

### X.25 Return and Reason Codes

The following table shows the return and reason codes that indicate data could not be received from the QOLRECV API.

**Note:** When these return and reason codes are returned, all output parameters except the return and reason codes will contain hexadecimal zeros.

Return / Reason Code	Meaning	Recovery
0/3203	No data available to be received.	Ensure that user-defined communications support has data available to be received before calling the QOLRECV API. Try the request again.
80/2200	Queue error detected. Escape message CPF91F1 will be sent to the application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Correct the error, enable the link, and try the request again.
80/2401	Input buffer or input buffer descriptor error detected. Escape message CPF91F1 will be sent to the application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Correct the error, enable the link, and try the request again.
80/3002	A previous error occurred on this link that was reported to the application program by escape message CPF91F0 or CPF91F1. However, the application program has attempted another operation.	Ensure the link is disabled and see messages in the job log for further information. If escape message CPF91F0 was sent to the application program, then report the problem using the ANZPRB command. Otherwise, correct the error, enable the link, and try the request again.
80/4000	Error recovery has been canceled for this link.	Ensure the link is disabled and see messages in the job log for further information. Correct the condition, enable the link, and try the request again.
80/9999	Internal system error detected. Escape message CPF91F0 will be sent to the application program when this return and reason code is received.	See messages in the job log for further information. Report the problem using the ANZPRB command.
83/3001	Link not enabled.	Correct the communications handle parameter. Try the request again.
83/3004	Link is enabling.	Wait for the enable-complete entry to be sent to the data queue or user queue. If the link was successfully enabled, try the request again.

### Return and Reason Codes for X.25 Operation X'0001'

Return / Reason Code	Meaning	Recovery
0/0	User data received successfully.	Continue processing.

### Return and Reason Codes for X.25 Operation X'B001'

Return / Reason Code	Meaning	Recovery
0/0	The X'B000' output operation was successful.	Continue processing.
83/1999	Incorrect data was specified in output buffer when the X'B000' output operation was issued.  <b>Note:</b> The data specified in the output buffer will be copied into the input buffer and the error offset field in the diagnostic data parameter will point to the incorrect data.	Correct the incorrect data. Then, try the X'B000' output operation again.
83/3204	Connection ending because a X'B100' output operation was issued.	Wait for notification of the completion of the X'B100' output operation from the QOLRECV API (X'B101' operation).
83/4001	Link failure, system starting error recovery for this link. The connection has ended.	Wait for the link to recover. Then, try the X'B000' output operation again.
83/4002	Connection failure. The connection has ended. The diagnostic data parameter will contain more information on this error.	Correct any errors and try the X'B000' output operation again.
83/4005	All SVC channels are currently in use, or the requested PVC channel is already in use.	Wait for a virtual circuit to become available. Then, try the X'B000' output operation again.

### Return and Reason Codes for X.25 Operation X'B101'

Return / Reason Code	Meaning	Recovery
0/0	The X'B100' output operation was successful. The connection has ended.	Continue processing.
83/1007	Connection identifier not valid because connection has already ended.	Continue processing.
83/1999	Incorrect data was specified in output buffer when the X'B100' output operation was issued.  <b>Note:</b> The data specified in the output buffer will be copied into the input buffer and the error offset field in the diagnostic data parameter will point to the incorrect data.	Correct the incorrect data. Then, try the X'B100' output operation again.

### Return and Reason Codes for X.25 Operation X'B111'

Return / Reason Code	Meaning	Recovery
0/0	The X'B100' output operation was successful. The connection has ended.	Continue processing.
83/1007	Connection identifier not valid because connection has already ended.	Continue processing.
83/3205	The X'B110' operation is rejected because the application has not received the X'B311' operation prior to requesting the X'B110' operation.	Correct the application.

### Return and Reason Codes for X.25 Operation X'B201'

Return / Reason Code	Meaning	Recovery
0/0	Incoming X.25 SVC call received successfully.	Continue processing.

### Return and Reason Codes for X.25 Operation X'B301'

Return / Reason Code	Meaning	Recovery
83/3201	The maximum amount of incoming user data that can be held by user-defined communications support for the application program on this connection has been exceeded.	Issue the X'B100' output operation to end the connection.
83/3202	A reset indication has been received on this connection. The X.25 cause and diagnostic code fields in the diagnostic data parameter will contain the cause and diagnostic codes of the reset indication.	Issue the X'BF00' output operation to send a reset confirmation packet.
83/4001	Link failure, system starting error recovery for this link.	Issue the X'B100' output operation to end the connection.
83/4002	Connection failure. The diagnostic data parameter will contain more information on this error.	Issue the X'B100' output operation to end the connection.

### Return and Reason Codes for X.25 Operation X'B311'

Return / Reason Code	Meaning	Recovery
83/4001	Link failure, system starting error recovery for this link. All connections that were active on this link are closed or cleared.	Issue the X'B110' operation to free the connections.

Return / Reason Code	Meaning	Recovery
83/4002	A network error has occurred that affects all connections on this link. All connections that were active on this link are closed or cleared. The diagnostic data contains more information on this error.	Issue the X'B110' operation to free the connections.

### Return and Reason Codes for X.25 Operation X'BF01'

Return / Reason Code	Meaning	Recovery
0/0	The X'BF00' output operation was successful. The diagnostic data parameter will contain information indicating if a reset request or reset confirmation packet was sent.	Continue processing.
83/1006	Operation not valid.	Do not issue the X'BF00' output operation on connections that do not support resets.
83/3201	The maximum amount of incoming user data that can be held by user-defined communications support for the application program on this connection has been exceeded.	Wait to receive a failure notification from the QOLRECV API indicating this condition (X'B301' operation, 83/3201 return and reason code). Then issue the X'B100' output operation to end the connection.
83/3204	Connection ending because a X'B100' output operation was issued.	Wait for notification of the completion of the X'B100' output operation from the QOLRECV API (X'B101' operation).
83/4001	Link failure, system starting error recovery for this link.	Wait to receive a failure notification from the QOLRECV API indicating this condition (X'B301' operation, 83/4001 return and reason code). Then, issue the X'B100' output operation to end the connection.
83/4002	Connection failure.	Wait to receive a failure notification from the QOLRECV API indicating this condition (X'B301' operation, 83/4002 return and reason code). Then, issue the X'B100' output operation to end the connection.

## Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF91F0 E	Internal system error.
CPF91F1 E	User-defined communications application error.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

[Top](#) | ["Communications APIs," on page 1](#) | [APIs by category](#)

---

## Send Data (QOLSEND) API

Required Parameter Group:	
1	Return code
<b>Output</b>	Binary(4)
2	Reason code
<b>Output</b>	Binary(4)
3	Diagnostic data
<b>Output</b>	Char(40)
4	New provider connection end point ID
<b>Output</b>	Binary(4)
5	New user end point connection ID
<b>Input</b>	Binary(4)
6	Existing provider connection end point ID
<b>Input</b>	Binary(4)
7	Communications handle
<b>Input</b>	Char(10)
8	Operation
<b>Input</b>	Char(2)
9	Number of data units
<b>Input</b>	Binary(4)
	Default Public Authority: *USE
Threadsafe: No	

The Send Data (QOLSEND) API performs output on a link that is currently enabled in the job in which the application program is running. The operation parameter allows you to specify the type of output operation to perform. The application program must provide the data associated with the output operation in the output buffer that was created when the link was enabled. For X'0000' operations, the application program must also provide a description of that data in the output buffer descriptor that was created when the link was enabled.

The types of output operations that can be performed on a link depend on the type of communications line that the link is using. See "LAN Output Operations" on page 42 for more information on output operations that are supported on links using a token-ring, Ethernet, wireless, or FDDI communications line. See "X.25 SVC and PVC Output Operations" on page 43 for more information on output operations that are supported on links using an X.25 communications line.

### Required Parameter Group

#### Return code

OUTPUT; BINARY(4)

The recovery action to take. See "Return and Reason Codes" on page 56.

#### Reason code

OUTPUT; BINARY(4)

The error that occurred. See "Return and Reason Codes" on page 56.

### Diagnostic data

OUTPUT; CHAR(40)

Additional diagnostic data. See “Diagnostic Data Parameter Format” on page 40 for more information.

The content of this parameter is only valid when the operation parameter is set to X'0000' or X'B400'.

### New provider connection end point ID

OUTPUT; BINARY(4)

The provider connection end point (PCEP) ID for the connection that is to be established. This identifier must be used on all subsequent calls to the QOLSEND API for this connection.

The content of this parameter is only valid for links using an X.25 communications line and when the operation parameter is set to X'B000'.

### New user connection end point ID

INPUT; BINARY(4)

The user connection end point (UCEP) ID for the connection that is to be established. This is the identifier on which all incoming data for this connection will be received. Any numeric value except zero should be used. See “Receive Data (QOLRECV) API” on page 21 for more information.

The content of this parameter is only valid for links using an X.25 communications line and when the operation parameter is set to X'B000' or X'B400'.

### Existing provider connection end point ID

INPUT; BINARY(4)

The PCEP ID for the connection on which this operation will be performed. For links using a token-ring, Ethernet, or wireless communications line, the content of this parameter must always be set to 1.

For links using an X.25 communications line, the content of this parameter is only valid when the operation parameter is set to X'0000', X'B100', X'B400', or X'BF00'. It must contain the PCEP ID that was returned in the new provider connection end point ID parameter from the call to the QOLSEND API with operation X'B000', or the PCEP ID that was returned in the new provider connection end point ID parameter from the call to the QOLRECV API with operation X'B201' (incoming call). See “Receive Data (QOLRECV) API” on page 21 for more information on receiving X.25 calls.

### Communications handle

INPUT; CHAR(10)

The name of the link on which to perform the output operation.

### Operation

INPUT; CHAR(2)

The type of output operation to perform. With the exception of X'0000', all values are only valid for links using an X.25 communications line. The valid values are as follows:

X'0000'	Send data.
X'B000'	Send call request packet (SVC) or open PVC connection.
X'B100'	Send clear packet (SVC) or close PVC connection.
X'B110'	Initiate final cleanup of all connections that were closed by the system.
	This operation is only valid when the application receives an X'B311' operation to receive connection failure data.
X'B400'	Send call accept packet (SVC).

X'BF00'

Send reset request packet or reset confirmation packet (SVC or PVC).

### Number of data units

INPUT; BINARY(4)

The number of data units in the output buffer that contain data. Any value between 1 and the number of data units created in the output buffer may be used.

The content of this parameter is only valid when the operation parameter is set to X'0000'.



**Note:** The number of data units created in the output buffer was returned in the data units created parameter on the call to the QOLELINK API. See "Enable Link (QOLELINK) API" on page 4 for more information.

## Diagnostic Data Parameter Format

The format of the diagnostic data parameter is shown below. The contents of the fields within this parameter are only valid on X'0000' and X'B400' operations for the indicated return and reason codes.

Field	Type	Description
Reserved	CHAR(2)	Reserved for extension.
Error code	CHAR(4)	Specifies hexadecimal diagnostic information that can be used to determine recovery actions. See "Error Codes" on page 299 for more information.  The content of this field is only valid for 83/4001, 83/4002, and 83/4003 return/reason codes.
Time stamp	CHAR(8)	The time the error occurred.  The content of this field is only valid for 83/4001, 83/4002, and 83/4003 return/reason codes.
Error log identifier	CHAR(4)	The hexadecimal identifier that can be used for locating error information in the error log.  The content of this field is only valid for 83/4001, 83/4002, and 83/4003 return/reason codes.
Reserved	CHAR(10)	Reserved for extension.



Field	Type	Description
Indicators	CHAR(1)	<p>Specifies indicators the user-defined communications application program can use for diagnosing a potential error condition. This is a bit sensitive field.</p> <p>The valid values for bit 0 (leftmost bit) are as follows:</p> <p>'0'B Either there is no message in the QSYSOPR message queue, or there is a message and it does not have the capability to run problem analysis report (PAR) to determine the cause of the error.</p> <p>'1'B There is a message in the QSYSOPR message queue for this error, and it does have the capability to run problem analysis report (PAR) to determine the cause of the error.</p> <p>The valid values for bit 1 are as follows:</p> <p>'0'B The line error can be retried.</p> <p>'1'B The line error cannot be retried.</p> <p>The valid values for bit 2 are as follows:</p> <p>'0'B The cause and diagnostic codes fields are not valid.</p> <p>'1'B The cause and diagnostic codes fields are valid.</p> <p>The valid values for bit 3 are as follows:</p> <p>'0'B The error has not been reported to the system operator message queue.</p> <p>'1'B The error has been reported to the system operator message queue.</p> <p>For example, consider the following values for the indicators field:</p> <p>X'20' A condition has caused X.25 cause and diagnostic codes to be passed to the application. This information can determine the cause of the condition.</p> <p>X'50' An error has occurred and been reported to the QSYSOPR message queue. The error cannot be retried.</p> <p>X'F0' An error has occurred and been reported to the QSYSOPR message queue. The error cannot be retried, and has X.25 cause and diagnostic codes associated with it. Also a problem analysis report can be generated to determine the probable cause.</p> <p>The content of this field is valid only for 83/4001, 83/4002, 83/3202 and 83/4003 return/reason codes.</p>
X.25 cause code	CHAR(1)	<p>Specifies additional information on the condition reported. See the X.25 Network Support  book for interpreting the values of this field.</p> <p>The content of this field is only valid for 83/4001, 83/4002 and 83/3202 return/reason codes.</p>
X.25 diagnostic code	CHAR(1)	<p>Specifies additional information on the condition reported. See the X.25 Network Support  book for interpreting the values of this field.</p> <p>The content of this field is only valid for 83/4001, 83/4002 and 83/3202 return/reason codes.</p>
Reserved	CHAR(1)	Reserved for extension.
Error offset	BINARY(4)	<p>The offset from the top of the output buffer to the incorrect data in the output buffer.</p> <p>The content of this field is only valid for a 83/1999 return/reason code.</p>
Reserved	CHAR(4)	Reserved for extension.

## LAN Output Operations

The only output operation supported on links using a token-ring, Ethernet, wireless, or FDDI communications line is X'0000' (send user data). For each data frame to be sent on the network, the application program must provide the following information:

- General LAN information, optional routing information, and user data in the next data unit of the output buffer, starting with the first data unit
- A description, in the corresponding element of the output buffer descriptor, of the information in that data unit.

For example, suppose a user-defined communications application program wants to send two data frames. The information for the first frame would be placed in first data unit of the output buffer and described in the first element of the output buffer descriptor. The information for the second frame would be placed in the second data unit of the output buffer and described in the second element of the output buffer descriptor. The number of data units parameter on the call to the QOLSEND API would be set to 2.

**Note:** The X'0000' operation is synchronous. Control will not return from the QOLSEND API until the operation completes.

### Data Unit Format-LAN Operation X'0000'

Each data frame to be sent on the network corresponds to a data unit in the output buffer. The information in each of these data units is made up of general LAN information, optional routing data, and user data.

Field	Type	Description
Length of general LAN information	BINARY(2)	The length of the general LAN information in the data unit. This must be set to 16.
Destination adapter address	CHAR(6)	Specifies, in packed form, the adapter address to which this data frame will be sent.  <b>Note:</b> Because user-defined communications support only allows connectionless service over LANs, it is not necessary for all frames being sent on a single output operation to have the same destination adapter address.
DSAP address	CHAR(1)	The service access point on which the destination system will receive this frame. Any value may be used.  <b>Note:</b> The Ethernet Version 2 standard does not use logical link control, which utilizes SAPs. Therefore, to send Ethernet Version 2 frames, a null DSAP address (X'00') must be specified in the DSAP address field. Also, the Ethernet Standard (ETHSTD) parameter in the Ethernet line description must be configured as either *ETHV2 or *ALL.
SSAP address	CHAR(1)	The service access point on which the iSeries server will send this frame. Any service access point configured in the token-ring, Ethernet, wireless, or FDDI line description may be used.  <b>Note:</b> The Ethernet Version 2 standard does not use logical link control, which utilizes SAPs. Therefore, to send Ethernet Version 2 frames, a null SSAP address (X'00') must be specified in the SSAP address field. Also, the Ethernet Standard (ETHSTD) parameter in the Ethernet line description must be configured as either *ETHV2 or *ALL.

Field	Type	Description
Access control	CHAR(1)	Specifies outbound frame priority and is mapped to the access priority bits in the access control field of 802.5 frames. For links using a token-ring communications line, any value between X'00' and X'07' may be used, where X'00' is the lowest priority and X'07' is the highest priority.  For links using an Ethernet or wireless communications line, the content of this field is not applicable and must be set to X'00'.
Priority control	CHAR(1)	Specifies how to interpret the value set in the access control field. For links using a token-ring communications line, the valid values are as follows:  X'00' Use any priority less than or equal to the value set in the access control field.  X'01' Use the priority exactly equal to the value set in the access control field.  X'FF' Use the iSeries server default priority.  For links using an Ethernet or wireless communications line, the content of this field is not applicable and must be set to X'00'.
Length of routing information	BINARY(2)	The length of the routing information in the data unit. For links using a token-ring communications line, any value between 0 and 18 may be used, where 0 indicates that there is no routing information.  For links using an Ethernet or wireless communications line, the content of this field is not applicable and must be set to 0 indicating that there is no routing information.
Length of user data	BINARY(2)	The length of the user data in the data unit. This must be less than or equal to the maximum frame size allowed on the service access point specified in the SSAP address field. See "Query Line Description (QOLQLIND) API" on page 10 to determine the maximum frame size allowed on the service access point specified in the SSAP address field.  For Ethernet Version 2 frames, this must be at least 48 and not more than 1502 (including 2 bytes for the Ethernet type field).  <b>Note:</b> Ethernet 802.3 frames will be padded when the user data is less than 46 bytes.

### Output Buffer Descriptor Element Format-LAN Operation X'0000'

The information specified in each data unit of the output buffer must be described in the corresponding element of the output buffer descriptor.

Field	Type	Description
Length	BINARY(2)	The number of bytes of information in the corresponding data unit of the output buffer. This must be equal to the length of the general LAN information plus the length of the routing information plus the length of the user data. See Format of the General LAN Information (page 25) in the Receive Data (QOLRECV) API for more information on the format of the general LAN information.
Reserved	CHAR(30)	Reserved for extension.

## X.25 SVC and PVC Output Operations

The following table shows the output operations that are supported on links using an X.25 communications line.

Operation	Meaning
X'0000'	Send user data (SVC or PVC).  <b>Note:</b> This is a synchronous operation. Control will not return from the QOLSEND API until the operation completes.
X'B000'	Send a call request packet (SVC) or open the PVC connection.  <b>Note:</b> This is an asynchronous operation. Notification of the completion of this operation will be returned from the QOLRECV API with operation X'B001' only after control returns from the QOLSEND API with a 0/0 return and reason code. See "Receive Data (QOLRECV) API" on page 21 for more information.
X'B100'	Send a clear packet (SVC) or close the PVC connection.  <b>Note:</b> This is an asynchronous operation. Notification of the completion of this operation will be returned from the QOLRECV API with operation X'B101' only after control returns from the QOLSEND API with a 0/0 return and reason code. See "Receive Data (QOLRECV) API" on page 21 for more information.
X'B110'	Close all connections which were cleared by the reason given in the connection failure data received on X'B311'.  <b>Note:</b> This is an asynchronous operation. Notification of the completion of this operation will be returned from the QOLRECV API with operation X'B111' only after control returns from the QOLSEND API with a 0/0 return and reason code. See "Receive Data (QOLRECV) API" on page 21 for more information.
X'B400'	Send a call accept packet (SVC only).  <b>Note:</b> This is a synchronous operation. Control will not return from the QOLSEND API until the operation completes.
X'BF00'	Send a reset request or reset confirmation packet (SVC or PVC).  <b>Note:</b> This is an asynchronous operation. Notification of the completion of this operation will be returned from the QOLRECV API with operation X'BF01' only after control returns from the QOLSEND API with a 0/0 return and reason code. See "Receive Data (QOLRECV) API" on page 21 for more information.
<b>Note:</b> The maximum number of outstanding asynchronous operations (notification of completion not yet received from the QOLRECV API) is five. All calls made to the QOLSEND API or QOLSETF API under this condition will be rejected with a return and reason code of 83/3200.	

## X.25 Operation X'0000'

This operation allows the application program to send user data on an SVC or PVC X.25 connection. The application must provide the following information:

- User data in the next data unit of the output buffer, starting with the first data unit
- A description, in the corresponding element of the output buffer descriptor, of the user data in that data unit.

For example, suppose a user-defined communications application program wants to send two data units of user data. The first portion of the user data would be placed in first data unit of the output buffer and described in the first element of the output buffer descriptor. The second portion of the user data would be placed in the second data unit of the output buffer and described in the second element of the output buffer descriptor. The number of data units parameter on the call to the QOLSEND API would be set to 2.

User-defined communications support automatically fragments the user data in each data unit into one or more appropriately sized X.25 packets based on the negotiated transmit packet size for the connection. All packets constructed for a data unit, except for the last (or only) packet, will always have the X.25 more data bit (M-bit) set on. See Output Buffer Descriptor Element Format-X.25 Operation X'0000' (page 45) for more information on how to set the X.25 M-bit on or off in the last (or only) packet constructed for a data unit.

## Data Unit Format-X.25 Operation X'000'

Each data unit in the output buffer consists solely of user data and starts offset 0 from the top of the data unit.

## Output Buffer Descriptor Element Format-X.25 Operation X'0000'

The user data specified in each data unit of the output buffer must be described in the corresponding element of the output buffer descriptor.

Field	Type	Description
Length	BINARY(2)	The number of bytes of user data in the corresponding data unit of the output buffer. This must always be less than or equal to the X.25 user data size parameter that was specified on the call to the QOLELINK API when the link was enabled. See "Enable Link (QOLELINK) API" on page 4 for more information.
More data indicator	CHAR(1)	Specifies whether the X.25 more data bit (M-bit) should be set on or off in the last (or only) X.25 packet constructed for the corresponding data unit. The valid values are as follows:  <i>X'00'</i> Set the M-bit off in the last (or only) X.25 packet constructed for the corresponding data unit.  <i>X'01'</i> Set the M-bit on in the last (or only) X.25 packet constructed for the corresponding data unit.  <b>Note:</b> When this value is selected, the length field must be set to a multiple of the negotiated transmit packet size for the connection.
Qualified data indicator	CHAR(1)	Specifies whether the X.25 qualifier bit (Q-bit) should be set on or off in all X.25 packets constructed for the corresponding data unit. The valid values are as follows:  <i>X'00'</i> Set the Q-bit off in all X.25 packets constructed for the corresponding data unit.  <i>X'01'</i> Set the Q-bit on in all X.25 packets constructed for the corresponding data unit.
Interrupt packet indicator	CHAR(1)	Specifies whether the user data in the corresponding data unit should be sent in an X.25 interrupt packet. The valid values are as follows:  <i>X'00'</i> Send the user data in the corresponding data unit in one or more X.25 data packets.  <i>X'01'</i> Send the user data in the corresponding data unit in an X.25 interrupt packet. An interrupt packet causes the data to be expedited.  <b>Note:</b> When this value is selected, the length field must be set to a value between 1 and 32, and the number of data units parameter on the call to the QOLSEND API must be set to 1. Also, the contents of the more data indicator, qualified data indicator, and delivery confirmation indicator fields are ignored.

Field	Type	Description
Delivery confirmation indicator	CHAR(1)	Specifies whether the X.25 delivery confirmation bit (D-bit) should be set on or off in all X.25 packets constructed for the corresponding data unit. The valid values are as follows:  <i>X'00'</i> Set the D-bit off in all X.25 packets constructed for the corresponding data unit.  <i>X'01'</i> Set the D-bit on in all X.25 packets constructed for the corresponding data unit.  <b>Note:</b> The iSeries server does not fully support delivery confirmation when sending user data. Confirmation is from the local data circuit equipment (DCE).
Reserved	CHAR(26)	Reserved for extension.

### X.25 Operation X'B000'

This operation allows the application program to either initiate an SVC call or to open a PVC connection. The application must provide the data for this operation in the first data unit of the output buffer. The output buffer descriptor is not used.

The format of the data required for the X'B000' operation depends on whether it is used to initiate an SVC call or to open a PVC connection. Each format is explained in the following table.

**Note:** When initiating an SVC call, the iSeries server chooses an available SVC to use. The logical channel identifier of the SVC that was chosen will be returned when notification of the completion of X'B000' is received from the QOLRECV API (operation X'B001'). See "Receive Data (QOLRECV) API" on page 21 for more information.

#### Data Unit Format-X.25 Operation X'B000' (Initiate an SVC Call)

The data for this operation starts at offset 0 from the top of the first data unit in the output buffer. The following table shows the format of the data required for the X'B000' operation when initiating an SVC call.

Field	Type	Description
Reserved	CHAR(1)	This field must be set to X'02'.
Reserved	CHAR(3)	This field must be set to hexadecimal zeros.
Transmit packet size	BINARY(2)	The requested transmit packet size for this connection. The valid values are 64, 128, 256, 512, 1024, 2048, and 4096. The value specified must be less than or equal to the transmit maximum packet size configured for this line. The special value of X'FFFF' may be specified to use the transmit default packet size configured for this line.  See "Query Line Description (QOLQLIND) API" on page 10 for information on determining the transmit maximum packet size and the transmit default packet size configured for this line.

Field	Type	Description
Transmit window size	BINARY(2)	<p>The requested transmit window size for this connection. The valid values are as follows:</p> <p>1-7        When modulus 8 is configured for this line.</p> <p>1-15      When modulus 128 is configured for this line.</p> <p>X'FFFF'   Use the transmit default window size configured for this line.</p> <p>See "Query Line Description (QOLQLIND) API" on page 10 for information on determining the modulus value and the transmit default window size configured for this line.</p>
Receive packet size	BINARY(2)	<p>The requested receive packet size for this connection. The valid values are 64, 128, 256, 512, 1024, 2048, and 4096. The value specified must be less than or equal to the receive maximum packet size configured for this line. The special value of X'FFFF' may be specified to use the receive default packet size configured for this line.</p> <p>See "Query Line Description (QOLQLIND) API" on page 10 for information on determining the receive maximum packet size and the receive default packet size configured for this line.</p>
Receive window size	BINARY(2)	<p>The requested receive window size for this connection. The valid values are as follows:</p> <p>1-7        When modulus 8 is configured for this line.</p> <p>1-15      When modulus 128 is configured for this line.</p> <p>X'FFFF'   Use the receive default window size configured for this line.</p> <p>See "Query Line Description (QOLQLIND) API" on page 10 for information on determining the modulus value and the receive default window size configured for this line.</p>
Reserved	CHAR(7)	This field must be set to hexadecimal zeros.
DTE address length	BINARY(1)	<p>The number of binary coded decimal (BCD) digits in the DTE address to call. The valid values are as follows:</p> <p>1-15      When extended network addressing is not configured for this line.</p> <p>1-17      When extended network addressing is configured in the line description.</p> <p>See "Query Line Description (QOLQLIND) API" on page 10 to determine if extended network addressing is configured for this line.</p>
DTE address	CHAR(16)	Specifies, in binary coded decimal (BCD), the DTE address to call. The address must be left justified and padded on the right with BCD zeros.
Reserved	CHAR(8)	This field must be set to hexadecimal zeros.
Delivery confirmation support	CHAR(1)	<p>Specifies whether the X.25 delivery confirmation bit (D-bit) should be set on or off in the call request packet. The valid values are as follows:</p> <p>X'00'      Set the D-bit off in the call request packet.</p> <p>X'01'      Set the D-bit on in the call request packet.</p>
Reserved	CHAR(7)	This field must be set to hexadecimal zeros.

Field	Type	Description
Closed user group indicator	CHAR(1)	Specifies whether the closed user group (CUG) identifier should be included in the call packet. The valid values are as follows:  X'00' Do not include the CUG identifier in the call packet. X'01' Include the CUG identifier in the call packet.
Closed user group identifier	CHAR(1)	The CUG identifier to be included in the call packet. The valid values are as follows:  X'00' When the closed user group indicator field is set to X'00' X'00'-X'99' When the closed user group indicator field is set to X'01'
Reverse charging indicator	CHAR(1)	Specifies reverse charging options. The valid values are as follows:  X'00' Do not request reverse charging. X'01' Request reverse charging.
Fast select indicator	CHAR(1)	Specifies fast select options. The valid values are as follows:  X'00' Do not request fast select. X'01' Request fast select with restriction. X'02' Request fast select without restriction.
X.25 facilities length	BINARY(1)	The number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be used.  <b>Note:</b> The iSeries server codes the closed user group, reverse charging, and fast select facilities in the X.25 facilities field, if the user requested them in the above fields. Additionally, if the network user identification parameter (NETUSRID) is specified in the line description, the network user identification (NUI) facility is coded in the field, following the other additional facilities, if present. Finally, if the packet and window size values specified are different than the network default, the facilities containing these values are coded in the field as well. The system will update the X.25 facilities length field appropriately for each facility to which the iSeries server adds the X.25 facilities field. This length cannot exceed 109 bytes.
X.25 facilities	CHAR(109)	Specifies additional X.25 facilities data requested.  <b>Note:</b> The application programmer should not code the facilities for NUI, fast select, reverse charging, closed user group, packet size, or window size in this field. By doing so, this field could contain duplicate facilities, which may not be consistently supported by all X.25 networks.
Reserved	CHAR(48)	This field must be set to hexadecimal zeros.
Call user data length	BINARY(2)	The number of bytes of data in the call user data field. The valid values are as follows:  0-16 When the fast select indicator field is set to X'00'. 0-128 When the fast select indicator field is set to X'01' or X'02'.
Call user data	CHAR(128)	The call user data.



Field	Type	Description
Reserved	CHAR(128)	This field must be set to hexadecimal zeros.
Control information	CHAR(1)	<p>Specifies control information for this connection. This is a bit-sensitive field with bit 0 (leftmost bit) defined for reset support. The remaining bits are undefined and should be set off ('0'B).</p> <p>The valid values for bit 0 are as follows:</p> <p>'0'B      Resets are not supported on this connection.</p> <p>            When this value is selected, the X'BF00' output operation will not be valid on this connection. Also, a reset indication packet received on this connection will cause the connection to be ended.</p> <p>'1'B      Resets are supported on this connection.</p> <p>            When this value is selected, the X'BF00' output operation will be valid on this connection. Also, the user-defined communications application program will be required to handle reset indications received on this connection.</p> <p>For example, consider the following values for the control information field:</p> <p>X'00'      Resets are not supported on this connection.</p> <p>X'80'      Resets are supported on this connection.</p>
Reserved	CHAR(3)	This field must be set to hexadecimal zeros.
Maximum data unit assembly size	BINARY(4)	<p>The maximum number of bytes of user data that is received in a complete X.25 packet sequence before passing the user data to the application. Any value between 1024 and 32767 may be used, and should be set to the largest value that the application will support.</p> <p><b>Notes:</b></p> <ol style="list-style-type: none"> <li>1. The system attempts to assemble the entire packet sequence before passing the data to the application. The only exception to this is when the size of the packet sequence exceeds the value the user specified for this field.</li> <li>2. If the number of bytes of user data received in a complete X.25 packet sequence is more than can fit into one data unit of the input buffer, the more data indicator field in the corresponding element of the input buffer descriptor will be set to X'01' and the remaining user data will be filled in the next data unit. See "Receive Data (QOLRECV) API" on page 21 for more information.</li> <li>3. There is no limitation on the number of bytes of user data that can be sent in a complete X.25 packet sequence. However, the QOLSEND API may need to be called more than once.</li> </ol>
Automatic flow control	BINARY(2)	<p>Relates to the amount of data that will be held by user-defined communications support before sending a receive not ready (RNR) packet to the sending system. The recommended value for this field is 32, but any value between 1 and 128 may be used.</p> <p><b>Note:</b> A receive ready (RR) packet will be sent when the user-defined communications application program receives some of the data.</p>
Reserved	CHAR(30)	This field must be set to hexadecimal zeros.

### Data Unit Format-X.25 Operation X'B000' (Open a PVC Connection)

The data for this operation starts at offset 0 from the top of the first data unit in the output buffer. The following table shows the format of the data required for the X'B000' operation when opening a PVC connection.

Field	Type	Description
Reserved	CHAR(1)	This field must be set to hexadecimal zeros.
Reserved	CHAR(1)	This field must be set to hexadecimal zeros.
Logical channel identifier	CHAR(2)	<p>The logical channel identifier of the PVC to open. Any PVC configured for this line that is eligible to be used by the network controller that the link is using may be specified and must be in the range of X'0001'-X'0FFF'.</p> <p>See "Query Line Description (QOLQLIND) API" on page 10 for information on determining the PVCs configured for this line that are eligible to be used by the network controller the link is using.</p>
Transmit packet size	BINARY(2)	<p>The requested transmit packet size for this connection. The valid values are 64, 128, 256, 512, 1024, 2048, and 4096. The value specified must be less than or equal to the transmit maximum packet size configured for this line. The special value of X'FFFF' may be specified to use the transmit default packet size configured for this line.</p> <p>See "Query Line Description (QOLQLIND) API" on page 10 for information on determining the transmit maximum packet size and the transmit default packet size configured for this line.</p>
Transmit window size	BINARY(2)	<p>The requested transmit window size for this connection. The valid values are as follows:</p> <p>1-7        When modulus 8 is configured for this line.</p> <p>1-15      When modulus 128 is configured for this line.</p> <p>X'FFFF'   Use the transmit default window size configured for this line.</p> <p>See "Query Line Description (QOLQLIND) API" on page 10 for information on determining the modulus value and the transmit default window size configured for this line.</p>
Receive packet size	BINARY(2)	<p>The requested receive packet size for this connection. The valid values are 64, 128, 256, 512, 1024, 2048, and 4096. The value specified must be less than or equal to the receive maximum packet size configured for this line. The special value of X'FFFF' may be specified to use the receive default packet size configured for this line.</p> <p>See "Query Line Description (QOLQLIND) API" on page 10 for information on determining the receive maximum packet size and the receive default packet size configured for this line.</p>
Receive window size	BINARY(2)	<p>The requested receive window size for this connection. The valid values are as follows:</p> <p>1-7        When modulus 8 is configured for this line.</p> <p>1-15      When modulus 128 is configured for this line.</p> <p>X'FFFF'   Use the receive default window size configured for this line.</p> <p>See "Query Line Description (QOLQLIND) API" on page 10 for information on determining the modulus value and the receive default window size configured for this line.</p>
Reserved	CHAR(32)	This field must be set to hexadecimal zeros.

Field	Type	Description
Delivery confirmation support	CHAR(1)	<p>The X.25 delivery confirmation bit (D-bit) support for this connection. The valid values are as follows:</p> <p><i>X'00'</i> D-bit will be supported for sending data but not for receiving data.</p> <p><b>Note:</b> When this value is selected and an X.25 packet is received with the D-bit set on, the input/output processor (IOP) will send a reset packet.</p> <p><i>X'01'</i> D-bit will be supported for sending data and for receiving data.</p>
Reserved	CHAR(427)	This field must be set to hexadecimal zeros.
Control information	CHAR(1)	<p>Specifies control information for this connection. This is a bit-sensitive field with bit 0 (leftmost bit) defined for reset support. The remaining bits are undefined and should be set off ('0'B).</p> <p>The valid values for bit 0 are as follows:</p> <p><i>'0'B</i> Resets are not supported on this connection.</p> <p>When this value is selected, the <i>X'BF00'</i> output operation will not be valid on this connection. Also, a reset indication packet received on this connection will cause the connection to be ended.</p> <p><i>'1'B</i> Resets are supported on this connection.</p> <p>When this value is selected, the <i>X'BF00'</i> output operation will be valid on this connection. Also, the user-defined communications application program will be required to handle reset indications received on this connection.</p> <p>For example, consider the following values for the control information field:</p> <p><i>X'00'</i> Resets are not supported on this connection.</p> <p><i>X'80'</i> Resets are supported on this connection.</p>
Reserved	CHAR(3)	This field must be set to hexadecimal zeros.
Maximum data unit assembly size	BINARY(4)	<p>The maximum number of bytes of user data that is received in a complete X.25 packet sequence before passing the user data to the application. Any value between 1024 and 32767 may be used, and should be set to the largest value that the application will support.</p> <p><b>Notes:</b></p> <ol style="list-style-type: none"> <li>1. The system attempts to assemble the entire packet sequence before passing the data to the application. The only exception to this is when the size of the packet sequence exceeds the value the user specified for this field.</li> <li>2. If the number of bytes of user data received in a complete X.25 packet sequence is more than can fit into one data unit of the input buffer, the more data indicator field in the corresponding element of the input buffer descriptor will be set to <i>X'01'</i> and the remaining user data will be filled in the next data unit. See "Receive Data (QOLRECV) API" on page 21 for more information.</li> <li>3. There is no limit of the number of bytes of user data that can be sent in a complete X.25 packet sequence. However, the QOLSEND API may need to be called more than once.</li> </ol>

Field	Type	Description
Automatic flow control	BINARY(2)	Relates to the amount of data that will be held by user-defined communications support before sending a receive not ready (RNR) packet to the sending system. The recommended value for this field is 32, but any value between 1 and 128 may be used.  <b>Note:</b> A receive ready (RR) packet will be sent when the user-defined communications application program receives some of the data.
Reserved	CHAR(30)	This field must be set to hexadecimal zeros.

## X.25 Operation X'B100'

This operation allows the application program to either send a clear packet on an SVC, close an SVC connection that was cleared by the remote system, or to close a PVC connection. The application must provide the data for this operation in the first data unit of the output buffer. The output buffer descriptor is not used.

The format of the data required for the X'B100' operation is the same whether or not it is used to send a clear packet on an SVC or to close a PVC connection. The format of the data required for the X'B100' operation should be set to hexadecimal zeros if it is used to close an SVC connection that was previously cleared by the remote system.

### Notes:

1. The iSeries server provides the confirmation of the clear indication, however, the local user-defined communications application must issue the X'B100' operation to free the PCEP for the connection.
2. Closing a PVC connection will cause a reset packet to be sent to the remote system.

### Data Unit Format-X.25 Operation X'B100'

The data for this operation starts at offset 0 from the top of the first data unit in the output buffer. The following table shows the format of the data required for the X'B100' operation.

Field	Type	Description
Reserved	CHAR(2)	This field must be set to hexadecimal zeros.
Cause code	CHAR(1)	The X.25 cause code.
Diagnostic code	CHAR(1)	The X.25 diagnostic code.
Reserved	CHAR(4)	This field must be set to hexadecimal zeros.
X.25 facilities length <sup>1</sup>	BINARY(1)	The number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be used.
X.25 facilities <sup>1</sup>	CHAR(109)	The X.25 facilities data.
Reserved	CHAR(48)	This field must be set to hexadecimal zeros.
Clear user data length <sup>1</sup>	BINARY(2)	The number of bytes of data in the clear user data field. Any value between 0 and 128 may be used.
Clear user data <sup>1</sup>	CHAR(128)	The clear user data.  <b>Note:</b> The CCITT standard recommends that this field only be present in conjunction with the fast select or call deflection selection facility. The iSeries server does not enforce this restriction, however.
Reserved	CHAR(216)	This field must be set to hexadecimal zeros.

<sup>1</sup> This field is not used for PVC connections and should be set to hexadecimal zeros.

## X.25 Operation X'B110'

This operation allows the application program to clean up all internal control information on all the connections over the link and free up all PCEPs and UCEPs. This operation is only valid following the receipt of the X'B311' operation that reports the connection failure data to the application. There is no data associated with this operation.

### X.25 Operation X'B400'

This operation allows the application program to accept an incoming SVC call. The application must provide the data for this operation in the first data unit of the output buffer. The output buffer descriptor is not used.

**Note:** Notification of incoming calls are received from the QOLRECV API with operation X'B201'. See "Receive Data (QOLRECV) API" on page 21 for more information.

### Data Unit Format-X.25 Operation X'B400'

The data for this operation starts at offset 0 from the top of the first data unit in the output buffer. The following table shows the format of the data required for the X'B400' operation.

Field	Type	Description
Reserved	CHAR(1)	This field must be set to hexadecimal zeros.
Reserved	CHAR(3)	This field must be set to hexadecimal zeros.
Transmit packet size	BINARY(2)	The transmit packet size for this connection. The valid values are 64, 128, 256, 512, 1024, 2048, and 4096. The value specified must be less than or equal to the transmit maximum packet size configured for this line. The special value of X'FFFF' may be specified to use the transmit default packet size configured for this line.  See "Query Line Description (QOLQLIND) API" on page 10 for information on determining the transmit maximum packet size and the transmit default packet size configured for this line.
Transmit window size	BINARY(2)	The transmit window size for this connection. The valid values are as follows:  1-7        When modulus 8 is configured for this line. 1-15      When modulus 128 is configured for this line. X'FFFF' Use the transmit default window size configured for this line.  See "Query Line Description (QOLQLIND) API" on page 10 for information on determining the modulus value and the transmit default window size configured for this line.
Receive packet size	BINARY(2)	The receive packet size for this connection. The valid values are 64, 128, 256, 512, 1024, 2048, and 4096. The value specified must be less than or equal to the receive maximum packet size configured for this line. The special value of X'FFFF' may be specified to use the receive default packet size configured for this line.  See "Query Line Description (QOLQLIND) API" on page 10 for information on determining the receive maximum packet size and the receive default packet size configured for this line.

Field	Type	Description
Receive window size	BINARY(2)	<p>The receive window size for this connection. The valid values are as follows:</p> <p>1-7        When modulus 8 is configured for this line.</p> <p>1-15      When modulus 128 is configured for this line.</p> <p>X'FFFF'   Use the receive default window size configured for this line.</p> <p>See "Query Line Description (QOLQLIND) API" on page 10 for information on determining the modulus value and the receive default window size configured for this line.</p>
Reserved	CHAR(32)	This field must be set to hexadecimal zeros.
Delivery confirmation support	CHAR(1)	<p>Specifies whether the X.25 delivery confirmation bit (D-bit) should be set on or off in the call accept packet. This also specifies the D-bit support for this connection. The valid values are as follows:</p> <p>X'00'      Set the D-bit off in the call accept packet. D-bit will be supported for sending data but not for receiving data.</p> <p><b>Note:</b> When this value is selected and an X.25 packet is received with the D-bit set on, the input/output processor (IOP) will send a reset packet.</p> <p>X'01'      Set the D-bit on in the call accept packet. D-bit will be supported for sending data and for receiving data.</p>
Reserved	CHAR(11)	This field must be set to hexadecimal zeros.
X.25 facilities length	BINARY(1)	<p>The number of bytes of data in the X.25 facilities field. Any value between 0 and 109 may be used.</p> <p><b>Note:</b> The iSeries server codes the packet and window size facilities in this field, if necessary. The total length of all facilities cannot exceed 109 bytes.</p>
X.25 facilities	CHAR(109)	<p>The X.25 facilities data.</p> <p><b>Note:</b> The application programmer should not code the facilities for packet or window sizes in this field. By doing so, this field could contain duplicate facilities, which may not be consistently supported by all X.25 networks.</p>
Reserved	CHAR(306)	This field must be set to hexadecimal zeros.
Control information	CHAR(1)	<p>Specifies control information for this connection. This is a bit-sensitive field with bit 0 (leftmost bit) defined for reset support. The remaining bits are undefined and should be set off ('0'B).</p> <p>The valid values for bit 0 are as follows:</p> <p>'0'B        Resets are not supported on this connection.</p> <p>            When this value is selected, the X'BF00' output operation will not be valid on this connection. Also, a reset indication packet received on this connection will cause the connection to be ended.</p> <p>'1'B        Resets are supported on this connection.</p> <p>            When this value is selected, the X'BF00' output operation will be valid on this connection. Also, the user-defined communications application program will be required to handle reset indications received on this connection.</p> <p>For example, consider the following values for the control information field:</p> <p>X'00'        Resets are not supported on this connection.</p> <p>X'80'        Resets are supported on this connection.</p>

Field	Type	Description
Reserved	CHAR(3)	This field must be set to hexadecimal zeros.
Maximum data unit assembly size	BINARY(4)	<p>The maximum number of bytes of user data that can be received in a complete X.25 packet sequence on this connection. If this limit is exceeded, the connection will be ended. Any value between 1024 and 32767 may be used.</p> <p><b>Notes:</b></p> <ol style="list-style-type: none"> <li>1. If the number of bytes of user data received in a complete X.25 packet sequence is more than can fit into one data unit of the input buffer, the more data indicator field in the corresponding element of the input buffer descriptor will be set to X'01' and the remaining user data will be filled in the next data unit. See "Receive Data (QOLRECV) API" on page 21 for more information.</li> <li>2. There is no limitation on the number of bytes of user data that can be sent in a complete X.25 packet sequence. However, the QOLSEND API may need to be called more than once.</li> </ol>
Automatic flow control	BINARY(2)	<p>Relates to the amount of data that will be held by user-defined communications support before sending a receive not ready (RNR) packet to the sending system. The recommended value for this field is 32, but any value between 1 and 128 may be used.</p> <p><b>Note:</b> A receive ready (RR) packet will be sent when the user-defined communications application program receives some of the data.</p>
Reserved	CHAR(30)	This field must be set to hexadecimal zeros.

## X.25 Operation X'BF00'

This operation allows an application program to send a reset request packet or a reset confirmation packet on an X.25 SVC or PVC connection. The application must provide the X.25 cause and diagnostic codes required for this operation in the first data unit of the output buffer. The output buffer descriptor is not used.

Information indicating whether a reset request or reset confirmation packet was sent is returned when notification of the completion of the X'BF00' operation is received from the QOLRECV API (operation X'BF01'). This information will be in the diagnostic data parameter of the QOLRECV API. See "Receive Data (QOLRECV) API" on page 21 for more information.

A reset confirmation packet will be sent under the following conditions:

- After a reset indication packet has been received on the connection and the application has received it from the QOLRECV API (X'B301' operation, 83/3202 return and reason code)
- After a reset indication packet has been received on the connection but before the application has received it from the QOLRECV API
- When a reset indication packet is received on the connection at the same time the X'BF00' output operation is issued

This is known as a reset collision. In this case, user-defined communications support will discard the reset indication and, therefore, the application program will not receive it from the QOLRECV API. However, the cause and diagnostic codes from the reset indication are returned in the diagnostic data parameter of the QOLRECV program when the application receives notification of the completion of the X'BF00' operation. See "Receive Data (QOLRECV) API" on page 21 for more information.

A reset request packet will be sent when none of the above conditions are true.

### Notes:

1. Data not yet received by the application program on a connection will *not* be deleted when a X'BF00' operation is issued on that connection. This data will be received before the notification of the

completion of the X'BF00' operation is received from the QOLRECV API (operation X'BF01'). Data received after the notification of the completion of the X'BF00' operation is received should be treated as new data.

2. The X'BF00' operation is only valid on connections that support resets. See X.25 Operation X'B000' (page 46) and X.25 Operation X'B400' (page 53) for more information on specifying reset support.

### Data Unit Format-X.25 Operation X'BF00'

The first 2 bytes of the data unit in the output buffer are used for this operation. The first byte contains the X.25 cause code. The second byte contains the X.25 diagnostic code.

## Return and Reason Codes

The return and reason codes that can be returned from the QOLSEND API depend on the type of communications line the link is using and on the operation that was requested.

### Return and Reason Codes for LAN Operation X'0000'

Return / Reason Code	Meaning	Recovery
0/0	Operation successful.	Continue processing.
80/2200	Queue error detected. Escape message CPF91F1 will be sent to the application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Then correct the error, enable the link, and try the request again.
80/2401	Output buffer or output buffer descriptor error detected. Escape message CPF91F1 will be sent to the application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Then correct the error, enable the link, and try the request again.
80/3002	A previous error occurred on this link that was reported to the application program by escape message CPF91F0 or CPF91F1. However, the application program has attempted another operation.	Ensure the link is disabled and see messages in the job log for further information. If escape message CPF91F0 was sent to the application program, then report the problem using the ANZPRB command. Otherwise, correct the error, enable the link, and try the request again.
80/4000	Error recovery has been canceled for this link.	Ensure the link is disabled and see messages in the job log for further information. Correct the condition, enable the link, and try the request again.
80/8000	The amount of user data in a data unit of the output buffer is greater than the maximum frame size allowed on the communications line the link is using. Escape message CPF91F1 will be sent to the application program when this return and reason code is received.	Ensure the link is disabled. Correct the error, enable the link, and try the request again.
80/9999	Internal system error detected. Escape message CPF91F0 will be sent to the application program when this return and reason code is received.	See messages in the job log for further information. Report the problem using the ANZPRB command.
83/1006	Output operation not valid.	Correct the operation parameter. Try the request again.
83/1007	Connection identifier not valid.	Correct the existing provider connection end point ID parameter. Try the request again.
83/1008	Number of data units not valid.	Correct the number of data units parameter. Try the request again.
83/1998	The amount of data in a data unit of the output buffer is not correct.	Correct the amount of user data, or the total amount of generalLAN information, routing information, and user data in the offending data unit. Try the request again.



Return / Reason Code	Meaning	Recovery
83/1999	Incorrect data in a data unit of the output buffer. The error offset field in the diagnostic data parameter will point to the incorrect data.	Correct the incorrect data. Try the request again.
83/3001	Link not enabled.	Correct the communications handle parameter. Try the request again.
83/3004	Link is enabling.	Wait for the enable-complete entry to be sent to the data queue or user queue. If the link was successfully enabled, try the request again.
83/4001	Link failure, system starting error recovery for this link.	Wait for the link to recover. Try the request again.
83/4003	Error detected by the input/output processor (IOP). The diagnostic data parameter will contain more information on this error.	Correct the error, and try the request again.

### General X.25 Return and Reason Codes

The following table shows the return and reason codes that can be received from the QOLSEND API for any requested operation.

Return / Reason Code	Meaning	Recovery
80/2200	Queue error detected. Escape message CPF91F1 will be sent to the application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Correct the error, enable the link, and try the request again.
80/2401	Output buffer or output buffer descriptor error detected. Escape message CPF91F1 will be sent to the application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Correct the error, enable the link, and try the request again.
80/3002	A previous error occurred on this link that was reported to the application program by escape message CPF91F0 or CPF91F1. However, the application has attempted another operation.	Ensure the link is disabled and see messages in the job log for further information. If escape message CPF91F0 was sent to the application program, report the problem using the ANZPRB command. Otherwise, correct the error, enable the link, and try the request again.
80/4000	Error recovery has been canceled for this link.	Ensure the link is disabled and see messages in the job log for further information. Correct the condition, enable the link, and try the request again.
80/9999	Internal system error detected. Escape message CPF91F0 will be sent to the application program when this return and reason code is received.	See messages in the job log for further information. Report the problem using the ANZPRB command.
83/1006	Output operation not valid.	Correct the operation parameter. Try the request again.
83/3001	Link not enabled.	Correct the communications handle parameter. Try the request again.
83/3004	Link is enabling.	Wait for the enable-complete entry to be sent to the data queue or user queue. If the link was successfully enabled, try the request again.
83/3200	All resources are currently in use by asynchronous operations that have not yet completed.	Wait for at least one of the asynchronous operations to complete. Notification of completion of these operations will be received from the QOLRECV API. Try the request again.

### Return and Reason Codes for X.25 Operation X'0000'

Return / Reason Code	Meaning	Recovery
0/0	Operation successful.	Continue processing.

Return / Reason Code	Meaning	Recovery
83/1007	Connection identifier not valid.	Correct the existing provider connection end point ID parameter. Try the request again.
83/1008	Number of data units not valid.	Correct the number of data units parameter. Try the request again.
83/1997	The amount of user data in a data unit of the output buffer is not a multiple of the negotiated transmit packet size, and the more data indicator in the corresponding element of the output buffer descriptor is set to X'01'.	Correct the amount of user data in the offending data unit. Try the request again.
83/1998	The amount of user data in a data unit of the output buffer is not correct.	Correct the amount of user data in the offending data unit. Try the request again.
83/3201	The maximum amount of incoming user data that can be held by user-defined communications support for the application program on this connection has been exceeded.	Wait to receive a failure notification from the QOLRECV API indicating this condition (X'B301' operation, 83/3201 return and reason code). Issue the X'B100' output operation to end the connection.
83/3202	A reset indication has been received on this connection. The X.25 cause and diagnostic code fields in the diagnostic data parameter will contain the cause and diagnostic codes of the reset indication.	Wait to receive notification from the QOLRECV API indicating this condition (X'B301' operation, 83/3202 return and reason code). Issue the X'BF00' output operation to send a reset confirmation packet.
83/3205	Connection not in a valid state.	Ensure the connection is in a valid state for this operation. Try the request again.
83/4001	Link failure, system starting error recovery for this link.	Wait to receive a failure notification from the QOLRECV API indicating this condition (X'B301' or X'B311' operation, 83/4001 return and reason code). Issue the X'B100' output operation to end the connection.
83/4002	Connection failure.	Wait to receive a failure notification from the QOLRECV API indicating this condition (X'B301' operation, 83/4002 return and reason code). Issue the X'B100' output operation to end the connection.
83/4003	Data not sent. Error detected by input/output processor.	Try the request again. If the error persists, use the ANZPRB command to analyze and report the problem.

### Return and Reason Codes for X.25 Operation X'B000'

Return / Reason Code	Meaning	Recovery
0/0	Operation initiated.	Wait for notification of the completion of the X'B000' operation from the QOLRECV API (X'B001' operation).
83/4005	All connections are currently in use.	Wait for a connection to become available and try the request again.

### Return and Reason Codes for X.25 Operation X'B100'

Return / Reason Code	Meaning	Recovery
0/0	Operation initiated.	Wait for notification of the completion of the X'B100' operation from the QOLRECV API (X'B101' operation).
83/1007	Connection identifier not valid.	Correct the existing provider connection end point ID parameter. Try the request again.
83/3205	Connection not in a valid state.	Ensure the connection is in a valid state for this operation. Try the request again.

## Return and Reason Codes for X.25 Operation X'B110'

Return / Reason Code	Meaning	Recovery
0/0	Operation initiated.	Wait for notification of the completion of the X'B110' operation from the QOLRECV API (X'B111' operation).

## Return and Reason Codes for X.25 Operation X'B400'

Return / Reason Code	Meaning	Recovery
0/0	Operation successful.	Continue processing.
83/1007	Connection identifier not valid.	Correct the existing provider connection end point ID parameter. Try the request again.
83/1999	Incorrect data in a data unit of the output buffer. The error offset field in the diagnostic data parameter will point to the incorrect data.	Correct the incorrect data. Try the request again.
83/3205	Connection not in a valid state.	Ensure the connection is in a valid state for this operation. Try the request again.
83/4001	Link failure, system starting error recovery for this link.	Issue the X'B100' output operation to end the connection.
83/4004	Inbound call timed out.	Issue the X'B100' output operation to end the connection.

## Return and Reason Codes for X.25 Operation X'BF00'

Return / Reason Code	Meaning	Recovery
0/0	Operation initiated.	Wait for notification of the completion of the X'BF00' operation from the QOLRECV API (X'BF01' operation).
83/1007	Connection identifier not valid.	Correct the existing provider connection end point ID parameter. Try the request again.
83/3205	Connection not in a valid state.	Ensure the connection is in a valid state for this operation. Try the request again.

## Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF91F0 E	Internal system error.
CPF91F1 E	User-defined communications application error.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

[Top](#) | ["Communications APIs," on page 1](#) | [APIs by category](#)

---

## Set Filter (QOLSETF) API

Required Parameter Group:	
1	Return code
<b>Output</b>	Binary(4)
2	Reason code
<b>Output</b>	Binary(4)
3	Error offset
<b>Output</b>	Binary(4)
4	Communications handle
<b>Input</b>	Char(10)
	Threadsafe: No

The Set Filter (QOLSETF) API activates and/or deactivates one or more filters for a link that is currently enabled in the job in which the application program is running. The application program must provide the required filter information in the output buffer that was created when the link was enabled. The output buffer descriptor is not used. See "Format of Filter Information" on page 61 for details on the format of the filter information in the output buffer.

Filters contain inbound routing information that user-defined communications support uses to route incoming data to a link that is enabled by an application program. The incoming data that is routed depends on the type of communications line the link is using. On an X.25 communications line, the incoming data is an incoming switched virtual circuit (SVC) call. On a token-ring, Ethernet, wireless, or FDDI communications line, the incoming data is the actual data frame.

The type of filters activated for a link determine the way incoming data is routed to that link.

**Note:** All active filters for a link must be of the same type.

For links using a token-ring, Ethernet, wireless, or FDDI communications line, there are three types of filters. The following list of filters is from most to least restrictive:

- Destination service access point (DSAP), source service access point (SSAP), frame type, optional sending adapter address, and protocol (or group) ID.
- Destination service access point (DSAP), source service access point (SSAP), optional frame type, and sending adapter address
- DSAP, SSAP, and optional frame type
- DSAP

For links using an X.25 communications line, there are two types of filters. The following list of filters is from most to least restrictive:

- Protocol identifier (PID) and calling data terminal equipment (DTE) address  
The iSeries server treats the first byte of call-user data in an X.25 call request packet as the PID.
- PID

The order for checking filters when multiple links are using the same communications line, is from most to least restrictive. For example, suppose two user-defined communications application programs (application program A and B) in different jobs each have a link enabled that use the same token-ring communications line. Further suppose that application program A has activated a filter on DSAP X'22' and application program B has activated a filter on DSAP X'22' and SSAP X'22'. If a data frame comes in

with a DSAP of X'22' and an SSAP of X'22', application program B will receive the frame. If a data frame comes in with a DSAP of X'22' and an SSAP not equal to X'22', application program A will receive the frame.

## Required Parameter Group

### Return code

OUTPUT: BINARY(4)

The recovery action to take. See "Return and Reason Codes" on page 67.

### Reason code

OUTPUT; BINARY(4)

The error that occurred. See "Return and Reason Codes" on page 67.

### Error offset

OUTPUT; BINARY(4)

The offset from the top of the output buffer to the incorrect filter header data or to the incorrect filter in the filter list.

The content of this parameter is only valid for 83/1999 and 83/3003 return/reason codes.

### Communications handle

INPUT; CHAR(10)

The name of the link on which to perform the filter operation.

## Format of Filter Information

The application must provide all filter information in the output buffer that was created when the link was enabled. The application should treat the output buffer as one large space with the size equal to the number of data units created for the output buffer multiplied by the size of each data unit. This information is returned by the QOLELINK API when the link was enabled.

The filter information in the output buffer is made up of two parts. The first portion starts at offset 0 from the top of the output buffer and contains filter header data. The second portion of the filter information starts immediately after the filter header data in the output buffer and contains the filters that make up the filter list.


### Filter Header Data

Field	Type	Description
Function	CHAR(1)	<p>The filter function to perform. The valid values are as follows:</p> <p>X'00' Deactivate all filters that are currently active for this link and activate the filters specified in the filter list for this link.</p> <p>X'01' Activate the filters specified in the filter list for this link. All filters currently active for this link will remain active.</p> <p>X'02' Deactivate the filters specified in the filter list that are currently active for this link.</p>

Field	Type	Description
Filter type	CHAR(1)	<p>The type of the filters in the filter list. All filters in the filter list must be of this type. In addition, this must be the same type as the filters currently active for this link, if any. The valid values are as follows:</p> <p><i>X'00'</i> PID. This filter type is only applicable for links using an X.25 communications line and only applies to incoming SVC calls.</p> <p><i>X'01'</i> PID and calling DTE address. This filter type is only applicable for links using an X.25 communications line and only applies to incoming SVC calls.</p> <p><i>X'02'</i> DSAP. This filter type is only applicable for links using a token-ring, Ethernet, wireless, or FDDI communications line.</p> <p><i>X'03'</i> DSAP, SSAP, and optional frame type. This filter type is only applicable for links using a token-ring, Ethernet, wireless, or FDDI communications line.</p> <p><i>X'04'</i> DSAP, SSAP, optional frame type, and sending adapter address. This filter type is only applicable for links using a token-ring, Ethernet, wireless, or FDDI communications line.</p> <p><i>X'08'</i> DSAP, SSAP, frame type, optional and sending adapter address, and protocol identifier (or organization ID). This filter type is only applicable for links using a LAN communications line.</p> <p><b>Note:</b> The filter type field must be set even if there are no filters in the filter list.</p>
Number of filters	BINARY(2)	<p>The number of filters in the filter list. Any value between 0 and 256 may be used.</p> <p><b>Note:</b> The maximum number of filters that can be specified in the filter list is also limited by the total size of the output buffer which may accommodate less than 256 filters.</p>
Filter length	BINARY(2)	<p>The length of each filter in the filter list. This value must be 16 for filter types <i>X'00'</i> and <i>X'01'</i>, and 14 for filter types <i>X'02'</i>, <i>X'03'</i>, and <i>X'04'</i>, and 25 for filter type <i>X'08'</i>.</p> <p><b>Note:</b> The filter length field must be set even if there are no filters in the filter list.</p>

The format of each filter in the previous list of filters is described in the following table. All filters in the list of filters must be contiguous with each other and be of the type specified in the filter type field in the filter header data.

### X.25 Filters (Filter Types *X'00'* and *X'01'*)

Field	Type	Description
PID length	CHAR(1)	<p>The length of the PID on which to route incoming calls. The valid values are as follows:</p> <p><i>X'00'</i> Route incoming calls with no PID specified. That is, with no call user data in the call request packet.</p> <p><i>X'01'</i> Route incoming calls with the PID being treated as the first byte of call user data in the call request packet.</p>
PID	CHAR(1)	<p>The PID on which to route incoming calls. This should be set to <i>X'00'</i> when the PID length field is set to <i>X'00'</i>. Otherwise, any value may be used.</p> <p><b>Note:</b> Care should be taken when setting the PID field to an SNA PID (<i>X'C3'</i>, <i>X'C6'</i>, <i>X'CB'</i>, <i>X'CE'</i>), asynchronous PID (<i>X'01'</i>, <i>X'C0'</i>), or TCP/IP PID (<i>X'CC'</i>). See the X.25 Network Support  book for more information.</p>
Calling DTE address length	CHAR(1)	<p>Specifies, in hexadecimal, the number of binary coded decimal (BCD) digits in the calling DTE address on which to route incoming calls. The valid values are as follows:</p> <p><i>X'00'</i> For filter type <i>X'00'</i>.</p> <p><i>X'01'</i>-<i>X'0F'</i> For filter type <i>X'01'</i> when extended network addressing is not configured in the line description. See "Query Line Description (QOLQLIND) API" on page 10 to determine if extended network addressing is configured for this line.</p> <p><i>X'01'</i>-<i>X'11'</i> For filter type <i>X'01'</i> when extended network addressing is configured in the line description. See "Query Line Description (QOLQLIND) API" on page 10 to determine if extended network addressing is configured for this line.</p>
Calling DTE address	CHAR(12)	<p>Specifies, in binary coded decimal (BCD), the calling DTE address on which to route incoming calls. This should be set to BCD zeros when the calling DTE address length field is set to <i>X'00'</i>. Otherwise, any valid DTE address left-justified and padded on the right with BCD zeros may be used.</p>

Field	Type	Description
Additional routing data	CHAR(1)	<p>Specifies additional data on which to route incoming calls. This field is applicable for all X.25 filter types and is bit-sensitive with bit 0 (leftmost bit) defined for reverse charging options and bit 1 defined for fast select options. The remaining bits are undefined and should be set off ('0'B).</p> <p>The valid values for bit 0 are as follows:</p> <p>'0'B      Accept reverse charging.</p> <p>'1'B      Do not accept reverse charging.</p> <p>The valid values for bit 1 are as follows:</p> <p>'0'B      Accept fast select.</p> <p>'1'B      Do not accept fast select.</p> <p>For example, consider the following values for the additional routing data field:</p> <p>X'00'      Accept reverse charging and accept fast select.</p> <p>X'40'      Accept reverse charging and do not accept fast select.</p> <p>X'80'      Do not accept reverse charging and accept fast select.</p> <p>X'C0'      Do not accept reverse charging and do not accept fast select.</p>

#### LAN Filters (Filter Types X'02', X'03', and X'04')

Field	Type	Description
DSAP address length	CHAR(1)	The length of the DSAP address on which to route incoming frames. This must be set to X'01'.
DSAP address	CHAR(1)	<p>The DSAP address on which to route incoming frames. The DSAP address is the service access point on which the incoming frame arrived. Any service access point configured in the token-ring, Ethernet, wireless, or FDDI line description as *NONSNA may be used.</p> <p><b>Note:</b> The Ethernet Version 2 standard does not use logical link control, which utilizes SAPs. Therefore, to receive Ethernet Version 2 frames, a null DSAP address (X'00') must be specified in the DSAP address field. Also, the Ethernet Standard (ETHSTD) parameter in the Ethernet line description must be configured as either *ETHV2 or *ALL.</p>
SSAP address length	CHAR(1)	<p>The length of the SSAP address on which to route incoming frames. The valid values are as follows:</p> <p>X'00'      For filter type X'02'.</p> <p>X'01'      For filter types X'03' and X'04'.</p>



Field	Type	Description
SSAP address	CHAR(1)	<p>The SSAP address on which to route incoming frames. The SSAP address is the service access point on which the incoming frame was sent. The valid values are as follows:</p> <p>X'00' For filter type X'02'.</p> <p>X'00'-X'FF' For filter types X'03' and X'04'.</p> <p><b>Note:</b> The Ethernet Version 2 standard does not use logical link control, which utilizes SAPs. Therefore, to receive Ethernet Version 2 frames, a null SSAP address (X'00') must be specified in the SSAP address field. Also, the Ethernet Standard (ETHSTD) parameter in the Ethernet line description must be configured as either *ETHV2 or *ALL.</p>
Frame type length	CHAR(1)	<p>The length of the frame type on which to route incoming frames. The valid values are as follows:</p> <p>X'00' For filter type X'02'. Also for filter types X'03' and X'04' when the DSAP address and SSAP address fields are not both set to X'00'.</p> <p>X'00' or X'02' For filter types X'03' and X'04' when the 'DSAP address' and SSAP address fields are both set to X'00'.</p>
Frame type	CHAR(2)	<p>The frame type on which to route incoming frames. The frame type is defined in an Ethernet Version 2 frame to indicate the upper layer protocol being used. This must be set to X'0000' when the frame type length field is set to X'00'. Otherwise, any value except X'80D5' (encapsulated LLC) may be used, but should be in the range of X'05DD'-X'FFFF'.</p>
Sending adapter address length	CHAR(1)	<p>Specifies, in hexadecimal, the length of the sending adapter address on which to route incoming frames. The valid values are as follows:</p> <p>X'00' For filter types X'02' and X'03'.</p> <p>X'06' For filter type X'04'.</p>
Sending adapter address	CHAR(6)	<p>Specifies, in packed form, the sending adapter address on which to route incoming frames. This must be set to X'000000000000' when the sending adapter address length field is set to X'00'. Otherwise, any valid adapter address may be used.</p>

### LAN Filters (Filter Type X'08')

Field	Type	Description
DSAP address length	CHAR(1)	<p>The length of the DSAP address on which to route incoming frames. This must be set to X'01'.</p>
DSAP address	CHAR(1)	<p>The DSAP address on which to route incoming frames. The DSAP address is the service access point on which the incoming frame arrived. Any service access point configured in the token-ring, Ethernet, wireless, or FDDI line description as *NONSNA may be used.</p> <p><b>Note:</b> The Ethernet Version 2 standard does not use logical link control, which utilizes SAPs. Therefore, to receive Ethernet Version 2 frames, a null DSAP address (X'00') must be specified in the DSAP address field. Also, the Ethernet Standard (ETHSTD) parameter in the Ethernet line description must be configured as either *ETHV2 or *ALL.</p>

Field	Type	Description
SSAP address length	CHAR(1)	The length of the SSAP address on which to route incoming frames. The valid values are as follows:  <i>X'01'</i> For filter type <i>X'08'</i> .
SSAP address	CHAR(1)	The SSAP address on which to route incoming frames. The SSAP address is the service access point on which the incoming frame was sent. The valid values are as follows:  <i>X'00'-X'FF'</i> For filter type <i>X'08'</i> .  <b>Note:</b> The Ethernet Version 2 standard does not use logical link control, which utilizes SAPs. Therefore, to receive Ethernet Version 2 frames, a null SSAP address ( <i>X'00'</i> ) must be specified in the SSAP address field. Also, the Ethernet Standard (ETHSTD) parameter in the Ethernet line description must be configured as either *ETHV2 or *ALL.
Frame type length	CHAR(1)	The length of the frame type on which to route incoming frames. The valid values are as follows:  <i>X'02'</i> For filter type <i>X'08'</i> .
Frame type	CHAR(2)	The frame type on which to route incoming frames. The frame type is defined in an Ethernet Version 2 frame to indicate the upper layer protocol being used. This must be set to <i>X'0000'</i> when the frame type length field is set to <i>X'00'</i> . Otherwise, any value except <i>X'80D5'</i> (encapsulated LLC) may be used, but should be in the range of <i>X'05DD'-X'FFFF'</i> .
Sending adapter address length	CHAR(1)	In hexadecimal, the length of the sending adapter address on which to route incoming frames. The valid values are as follows:  <i>X'00'</i> or <i>X'06'</i> For filter type <i>X'08'</i> .
Sending adapter address	CHAR(6)	In packed form, the sending adapter address on which to route incoming frames. This must be set to <i>X'000000000000'</i> when the sending adapter address length field is set to <i>X'00'</i> . Otherwise, any valid adapter address may be used.
Protocol ID length	CHAR(1)	In hexadecimal, the length of the protocol ID on which to route incoming frames. This must be set to <i>X'03'</i> .
Protocol ID	CHAR(3)	In hexadecimal, the protocol ID (or organization ID) to route incoming frames.
Reserved field	CHAR(7)	This field must be initialized to hexadecimal zeros, <i>X'0000000000000000'</i> .

## General Rules for Using Filters

The following is a list of rules for activating and deactivating filters:

- All active filters for a link must be of the same type
- A link can have a maximum of 256 active filters

- The maximum number of filters that can be specified in the filter list can be no more than 256, and may be less, depending on the size of the output buffer
- A request to activate a filter for a link that already has the same filter active will be successful, but the filter will only be activated once
- A request to deactivate a filter for a link that has no such filter active will be successful
- If the return and reason code from the QOLSETF API is not 0/0, none of the specified filters were activated or deactivated
- Once a filter is activated, it will remain active until one of the following occurs:
  - It is deactivated by explicitly calling the QOLSETF API
  - The link that the filter was active for is disabled

## Return and Reason Codes

Return/Reason Code	Meaning	Recovery
0/0	Operation successful.	Continue processing.
80/2200	Queue error detected. Escape message CPF91F1 will be sent to the application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Then correct the error, enable the link, and try the request again.
80/2401	Output buffer error detected. Escape message CPF91F1 will be sent to the application program when this return and reason code is received.	Ensure the link is disabled and see messages in the job log for further information. Then correct the error, enable the link, and try the request again.
80/3002	A previous error occurred on this link that was reported to the application program by escape message CPF91F0 or CPF91F1. However, the application program has attempted another operation.	Ensure the link is disabled and see messages in the job log for further information. If escape message CPF91F0 was sent to the application program, then report the problem using the ANZPRB command. Otherwise, correct the error, enable the link, and try the request again.
80/4000	Error recovery has been canceled for this link.	Ensure the link is disabled and see messages in the job log for further information. Then correct the condition, enable the link, and try the request again.
80/9999	Internal system error detected. Escape message CPF91F0 will be sent to the application program when this return and reason code is received.	See messages in the job log for further information. Then, report the problem using the ANZPRB command.
83/1998	The size of the output buffer is not large enough for the specified number of filters.	Reduce the number of filters in the filter list so that the size of the filter list plus the size of the filter header data is less than or equal to the size of the output buffer. Try the request again.
83/1999	Incorrect filter header data or incorrect filter in the filter list. If the filter header data is incorrect, the error offset parameter will point to the field in error. If a filter in the filter list is incorrect, the error offset parameter will point to the beginning of the incorrect filter.	Correct the incorrect filter header data or the incorrect filter in the filter list. Try the request again.
83/3001	Link not enabled.	Correct the communications handle parameter. Try the request again.

Return/Reason Code	Meaning	Recovery
83/3003	<p>One of the following is true of a filter in the filter list. The error offset parameter will point to the beginning of the offending filter.</p> <ul style="list-style-type: none"> <li>The filter is already activated by another job using the same communications line</li> <li>The service access point, specified in the DSAP address field of the filter, is not configured in the token-ring, Ethernet, wireless, or FDDI line description</li> <li>The DSAP address field of the filter contains the null DSAP address (X'00'), but the Ethernet Standard (ETHSTD) parameter in the Ethernet line description is not configured as *ETHV2 or *ALL</li> <li>The service access point, specified in the DSAP address field of the filter, is configured in the token-ring, Ethernet, wireless, or FDDI line description for SNA use only (*SNA)</li> </ul>	<p>Do one of the following, and try the request again:</p> <ul style="list-style-type: none"> <li>End the job that has already activated the filter</li> <li>Configure the service access point in the token-ring, Ethernet, wireless, or FDDI line description</li> <li>Delete the Ethernet line description, and create another Ethernet line description specifying *ETHV2 or *ALL in the Ethernet Standard (ETHSTD) parameter</li> <li>Change the service access point in the token-ring, Ethernet, or wireless line description to non-SNA use (*NONSNA)</li> </ul>
83/3004	Link is enabling.	Wait for the enable-complete entry to be sent to the data queue or user queue. If the link was successfully enabled, try the request again.
83/3200	<p>All resources are currently in use by asynchronous operations that have not yet completed.</p> <p><b>Note:</b> This return and reason code is only possible for links using an X.25 communications line. See "Send Data (QOLSEND) API" on page 38 for more information.</p>	Wait for at least one of the asynchronous operations to complete. Notification of completion of these operations will be received from the QOLRECV API. Try the request again.
83/4001	Link failure, system starting error recovery for this link.	Wait for the link to recover. Try the request again.

## Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF91F0 E	Internal system error.
CPF91F1 E	User-defined communications application error.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

[Top](#) | ["Communications APIs," on page 1](#) | [APIs by category](#)

## Set Timer (QOLTIMER) API

Required Parameter Group:	
1	Return code
<b>Output</b>	Binary(4)
2	Reason code
<b>Output</b>	Binary(4)
3	Timer set
<b>Output</b>	Char(8)
4	Timer to cancel
<b>Input</b>	Char(8)
5	Qualified queue name
<b>Input</b>	Char(20)
6	Operation
<b>Input</b>	Char(1)
7	Interval
<b>Input</b>	Binary(4)
8	Establish count
<b>Input</b>	Binary(4)
9	Key length
<b>Input</b>	Binary(4)
10	Key value
<b>Input</b>	Char(256)
11	User data
<b>Input</b>	Char(60)
Optional Parameter:	
12	Queue type
<b>Input</b>	Char(1)
Threadsafe: No	

The Set Timer (QOLTIMER) API either sets or cancels a timer. Up to 128 timers, each uniquely identified by a name (timer handle), can be set in the job in which the application program is running.

When the QOLTIMER API is called to set a timer, a timer handle (timer set parameter) is returned to the application program. The timer handle, along with the user data supplied when the timer was set, is included in the timer-expired entry that is sent to the data queue or user queue when the specified amount of time for this timer has elapsed. The timer is then reestablished, if necessary.

For example, suppose a user-defined communications application program sets a timer with a five-second interval to be established two times. After five seconds, the timer-expired entry for this timer will be sent to the data queue or user queue specified when the timer was set. The timer will then be automatically reestablished, and five seconds later, another timer-expired entry for this timer will be sent to the data queue or user queue. See "Timer-Expired Entry" on page 288 for the format of the timer-expired entry.

In addition to setting a timer, the application program can call the QOLTIMER API to cancel one or all timers currently set in the job in which the application program is running. User-defined communications support will implicitly cancel a timer in the following cases:

- After a timer has expired the specified number of times (establish count parameter)
- When a job ends that had one or more timers set

**Note:** User-defined communications support does not associate timers with links. If necessary, that association must be done by the application.

## Required Parameter Group

### Return code

OUTPUT; BINARY(4)

The recovery action to take. See “Return and Reason Codes” on page 71.

### Reason code

OUTPUT; BINARY(4)

The error that occurred. See “Return and Reason Codes” on page 71.

### Timer set

OUTPUT; CHAR(8)

The name of the timer (timer handle) that was set. TIMER001, TIMER002, ... , TIMER128 are the possible values.

The content of this parameter is only valid when setting a timer.

### Timer to cancel

INPUT; CHAR(8)

The name of the timer (timer handle) to cancel. TIMER001, TIMER002, ... , TIMER128 may be used as values. The special value of \*ALL (left-justified and padded on right with spaces) may be used to cancel all timers currently set in the job in which the user-defined communications application program is running.

The content of this parameter is only valid when canceling a timer.

### Qualified queue name

INPUT; CHAR(20)

The name and library of the data queue or user queue where the timer-expired entry will be sent when the timer expires. The first 10 characters specify the name of the data queue or user queue and the second 10 characters specify the library in which the queue is located. Both entries are left-justified. The special values of \*LIBL and \*CURLIB may be used for the library name.

The content of this parameter is only valid when setting a timer.

### Operation

INPUT; CHAR(1)

The timer operation to perform. The valid values are as follows:

X'01'	Set a timer.
X'02'	Cancel a timer.

### Interval

INPUT; BINARY(4)

The number of milliseconds for which to set this timer. Any value between 1,048 and 3,600,000 may be used.

The content of this parameter is only valid when setting a timer.

**Establish count**

INPUT; BINARY(4)

The number of times this timer will be established. Any value between 1 and 60 may be used. The special value of -1 may be used to always have this timer established after it expires.

The content of this parameter is only valid when setting a timer.

**Key length**

INPUT; BINARY(4)

The key length when using a keyed data queue or user queue. Any value between 0 and 256 may be used, where 0 indicates the data queue or user queue is not keyed.

The content of this parameter is only valid when setting a timer.

**Key value**

INPUT; CHAR(256)

The key value when using a keyed data queue or user queue.

The content of this parameter is only valid when setting a timer.

**User data**

INPUT; CHAR(60)

The user data that is to be included in the timer-expired entry when the timer expires.

The content of this parameter is only valid when setting a timer.

**Note:** This data is treated as character data only and should not contain pointers.

**Optional Parameter****Queue type**

INPUT; CHAR(1)

The type of queue you specified for the queue name parameter.

*D* Data queue

*U* User queue

**Return and Reason Codes**

Return/Reason Code	Meaning	Recovery
0/0	Operation successful.	Continue processing.
81/9999	Internal system error detected. Escape message CPF91F0 will be sent to the application program when this return and reason code is received.	See messages in the job log for further information. Report the problem using the ANZPRB command.
82/1011	Queue type not valid.	Correct the queue type parameter. Try the request again.
83/1001	Key length not valid.	Correct the key length parameter. Try the request again.
83/1009	Timer operation not valid.	Correct the operation parameter. Try the request again.
83/1010	Timer interval not valid.	Correct the interval parameter. Try the request again.
83/1011	Number of times to establish timer not valid.	Correct the establish count parameter. Try the request again.
83/3400	Timer not valid on cancel operation.	Correct the timer to cancel parameter. Try the request again.

Return/Reason Code	Meaning	Recovery
83/3401	All timers are currently set for the requested set operation.	Cancel a timer. Try the request again.
83/3402	Timer not set on cancel operation.	Continue processing.

## Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF91F0 E	Internal system error.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R1

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

---

## Data Stream Translation APIs

The data stream translation APIs allow your user-written applications access to the data stream translation routines for 5250, 3270, and formatted buffer display data streams. Only display device data streams are supported by these APIs. For more information on display data streams using formatted

buffers, see the SNA Upline Facility Programming  book.

For additional information, see Using the Data Stream APIs.

The data stream translation APIs are:

- “End Data Stream Translation Session (QD0ENDTS) API” (QD0ENDTS) ends a session for data stream translation.
- “Start Data Stream Translation Session (QD0STRTS) API” on page 73 (QD0STRTS) starts a session for data stream translation.
- “Translate Data Stream (QD0TRNDS) API” on page 76 (QD0TRNDS) translates a data stream in one format to another format.

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

---

## End Data Stream Translation Session (QD0ENDTS) API

Required Parameter Group:	
1	Translation session handle
<b>Input</b>	Char(16)
2	Error code
<b>I/O</b>	Char(*)
	Threadsafe: No

The End Data Stream Translation Session (QD0ENDTS) API ends a session for data stream translation.



## Required Parameter Group

### Translation session handle

INPUT; CHAR(16)

The name of the translation session. This name is returned to your application following the call to the QD0STRTS API.

### 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
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF5D58 E	Translation session handle parameter value not valid.
CPF5D67 E	Severe error occurred while addressing parameter list.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

---

## Start Data Stream Translation Session (QD0STRTS) API

Required Parameter Group:	
<b>1</b>	Translation session handle
<b>Output</b>	Char(16)
<b>2</b>	Display device name
<b>Input</b>	Char(10)
<b>3</b>	Default screen size
<b>Input</b>	Char(10)
<b>4</b>	Alternate screen size
<b>Input</b>	Char(10)
<b>5</b>	Error code
<b>I/O</b>	Char(*)
	Threadsafe: No

The Start Data Stream Translation Session (QD0STRTS) API initiates a session for data stream translation. Your application can start as many translation sessions as you need.

## Authorities and Locks

### *Device Authority*

The user must have at least \*USE authority to the device specified in the display device name parameter.

## Required Parameter Group

### Translation session handle

OUTPUT; CHAR(16)

The name of the translation session. This name is supplied to your application so that you can keep track of a particular session. It is also required that you pass this name to the other data stream APIs.

### Display device name

INPUT; CHAR(10)

The name of the 5250 device for which the translation is being done. The 5250 data stream that is generated depends on the capabilities of the display device. You can specify the following values:

<i>Name</i>	The name of a display device that is known to the system.  <b>Note:</b> An error will occur if the job you are using for data stream translation is not authorized to the device you specify. The display device that is associated with this job is to be used.
<i>*REQUESTER</i>	<b>Note:</b> An error will occur if there is no display device associated with this job. For example, the job is a batch job. The display device is assumed to have the lowest common characteristics. The following characteristics are assumed:
<i>*BASIC</i>	<ul style="list-style-type: none"><li>• The display is monochrome.</li><li>• The display has a screen size of 24x80. If a larger screen size is specified when *BASIC is specified for the display device name, an error occurs.</li><li>• Input in row 1, column 1 is not supported.</li><li>• The Home key does not work like the 3270 home key.</li><li>• The maximum number of input fields is 126.</li><li>• The language is defaulted to the Keyboard Type (QKBDTYPE) system value.</li><li>• The display does not support extended attributes.</li></ul>

**Note:** The full capabilities of the device can be determined only if a 5250 query has been sent to the device. The 5250 query is sent the first time a user signs on after the device is varied on. The results remain in effect until the device is varied off. If no one has signed on since the device was varied on, some of the characteristics will default to those assumed for \*BASIC display devices.

### Default screen size

INPUT; CHAR(10)

The size of the screen for the selected display device. Either this value or the alternate screen size value is used depending on the command used in the 3270 data stream. The possible screen sizes are:

<i>024X080</i>	24 lines by 80 columns
<i>027X132</i>	27 lines by 132 columns
<i>*DEVMAX</i>	The maximum screen size allowed by the device

### Alternate screen size

INPUT; CHAR(10)

The alternate size of the screen for the selected display device. Either this value or the default screen size value is used depending on the command used in the 3270 data stream. The possible screen sizes are:

<i>024X080</i>	24 lines by 80 columns
<i>027X132</i>	27 lines by 132 columns

\*DEVMAX        The maximum screen size allowed by the device

### 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
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF5D50 E	Display device description &1 not found.
CPF5D51 E	Device &1 is not a display device.
CPF5D52 E	Not authorized to display device &1.
CPF5D5B E	Value &1 for default screen size parameter not valid.
CPF5D61 E	Value for display device parameter not valid.
CPF5D66 E	Value for alternate screen size parameter not valid.
CPF5D67 E	Severe error occurred while addressing parameter list.
CPF5D68 E	Default screen size parameter is not valid.
CPF5D69 E	Alternate screen size parameter is not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

---

## Translate Data Stream (QD0TRNDS) API

Required Parameter Group:	
1	Translation session handle
<b>Input</b>	Char(16)
2	To buffer
<b>Output</b>	Char(*)
3	To buffer output length
<b>Output</b>	Binary(4)
4	To buffer length
<b>Input</b>	Binary(4)
5	To buffer type
<b>Input</b>	Char(10)
6	From buffer
<b>Input</b>	Char(*)
7	From buffer length
<b>Input</b>	Binary(4)
8	From buffer type
<b>Input</b>	Char(10)
9	Operation
<b>Input</b>	Char(1)
10	Error code
<b>I/O</b>	Char(*)
	Default Public Authority: *USE
Threadsafe: No	

The Translate Data Stream (QD0TRNDS) API translates data from one format to another format. The data formats depend on the parameter values you specify.

### Required Parameter Group

#### Translation session handle

INPUT; CHAR(16)

The name of the translation session. This name is returned to your application following the call to the QD0STRTS API.

#### To buffer

OUTPUT; CHAR(\*)

The buffer used to contain the output of the data stream translation. This value should be large enough to contain the expected results.

#### To buffer output length

OUTPUT; BINARY(4)

The length of the translated data that is placed in the to buffer parameter.

**To buffer length**

INPUT; BINARY(4)

The length of the buffer that is available for output.

**To buffer type**

INPUT; CHAR(10)

The type of data to be put into the to buffer parameter. The possible values are:

5250	Create a 5250 data stream
3270	Create a 3270 data stream
3270RB	Create a 3270 data stream for the data stream that is expected in response to a 3270 Read Buffer command
*FORMAT	Create a formatted buffer for the data. See the <i>SNA Upline Facility Programming</i> , SC41-5446, book to determine the format of the buffer header.

See Valid Parameter Combinations (page 78) for a list of the allowable combinations of this parameter with the operations and from buffer type parameters.

**From buffer**

INPUT; CHAR(\*)

The buffer that contains the data to be translated.

**From buffer length**

INPUT; BINARY(4)

The length of the data contained in the from buffer parameter.

**From buffer type**

INPUT; CHAR(10)

The type of data that is contained in the from buffer parameter. The possible values are:

5250	Contains a 5250 data stream
5250RS	Contains a 5250 data stream that results from a 5250 Read Screen command
5250RSE	Contains a 5250 data stream that results from a 5250 Read Screen with Extended Attributes command
3270	Contains a 3270 data stream
*FORMAT	Contains a formatted buffer for the data. See the <i>SNA Upline Facility Programming</i> book, SC41-5446, to determine the format of the buffer header.

See Valid Parameter Combinations (page 78) for a list of the allowable combinations of this parameter with the operations and to buffer type parameters.

**Operation**

INPUT; CHAR(1)

Indicates whether the data to be translated is input or output data. You can specify the following values:

I	The data to be translated is for an input operation
O	The data to be translated is for an output operation

See Valid Parameter Combinations (page 78) for a list of the allowable combinations of this parameter with the to buffer type and from buffer type parameters.

**Error code**

I/O; CHAR(\*)

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

The following table lists the valid combinations of the from buffer type, to buffer type, and operations parameters.

### Valid Parameter Combinations

Operation	From BufferType	To BufferType
O	3270	*FORMAT
O	3270	5250
O	*FORMAT	5250
I	5250	*FORMAT
I	5250	3270
I	*FORMAT	3270
I	5250RS	3270RB
I	5250RSE	3270RB

## Error Messages

Message ID	Error Message Text
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF5D53 E	To and from buffers overlap.
CPF5D54 E	Value &1 for operation parameter not valid.
CPF5D55 E	Value &1 is not valid for the To buffer type parameter.
CPF5D56 E	Value &1 is not valid for the From buffer type parameter.
CPF5D57 E	Combination of parameter values not valid.
CPF5D58 E	Translation session handle parameter value not valid.
CPF5D59 E	Value &1 for from buffer length parameter not valid.
CPF5D5A E	Value &1 for the to buffer length parameter not valid.
CPF5D5C E	3270 data stream in from buffer not valid.
	An error was found while translating the 3270 data stream in the from buffer. The error code for translation was &1.
X'0002'	A 3270 command or order that is not supported or not valid was detected in the data stream.
X'0003'	A parameter or address that is not valid was detected in the 3270 data stream.
X'0004'	Excess fields were detected in the data stream. A certain number of these fields are allowed based on the device specified on the QD0STRTS call. This number of fields was exceeded.
X'0021'	A set buffer address order is missing after a row-column AID code.
X'0863'	A character set attribute that is not valid was found in the data stream.

Message ID	Error Message Text
CPF5D5D E	5250 data stream in from buffer not valid.  An error was found while translating the 5250 data stream in the from buffer. The error code for the translation was &1.  X'0001' A 5250 AID code that was not correct was found in the data stream. X'0020' A cursor position that was not valid was detected in the 5250 data stream. X'0021' A set buffer address order is missing after a row-column AID code. X'0022' A set buffer address order that was not valid was found in the data stream. X'D030' A data stream resulting from a Read Screen with Extended Attributes command was specified for a display device that does not support extended attributes.
CPF5D5E E	Return code in formatted buffer indicates error. Codes returned in this message are listed in <i>SNA Upline Facility Programming</i> , SC41-5446.
CPF5D5F E	Data integrity error in from buffer. The error code for the translation was &1. The possible error codes are:  X'0023' Character not valid. X'0050' Shift out (X'0E') and shift in (X'0F') not correctly balanced in a DBCS session. X'0051' Shift out (X'0E') and shift in (X'0F') in a DBCS field. X'0052' The dead position in a DBCS field is not null. X'0053' A DBCS character is not valid.
CPF5D60 E	To buffer not large enough for translation output.
CPF5D62 E	Error occurred in translation routines.
CPF5D63 E	Data integrity error in formatted buffer. The error code for the translation was &1. The possible error codes are:  X'0023' Character not valid. X'0050' Shift out (X'0E') and shift in (X'0F') not correctly balanced in a DBCS session. X'0051' Shift out (X'0E') or shift in (X'0F') in a DBCS field. X'0052' The dead position in a DBCS field is not null. X'0053' A DBCS character is not valid.
CPF5D64 E	To buffer length not valid for to buffer.
CPF5D65 E	From buffer length not valid for from buffer.
CPF5D67 E	Severe error occurred while addressing parameter list.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V2R2

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

---

## OptiConnect APIs

The OptiConnect APIs are used to move user data between two or more servers that are connected by the OptiConnect fiber-optic bus. The OptiConnect APIs require that the OptiConnect hardware and software products have been installed on all of the systems that will be used for communications. A maximum of 32KB (where KB equals 1024 bytes) of data may be transferred in a single send or receive function.

**Note:** To use these APIs, you need the OptiConnect for OS/400<sup>(R)</sup> feature.

The OptiConnect APIs are:

- “Close Path (QzdmClosePath) API” (QzdmClosePath) closes an OptiConnect path.
- “Close Stream (QzdmCloseStream) API” on page 82 (QzdmCloseStream) closes an OptiConnect stream.
- “Open Path (QzdmOpenPath) API” on page 84 (QzdmOpenPath) opens an OptiConnect path.
- “Open Stream (QzdmOpenStream) API” on page 87 (QzdmOpenStream) opens an OptiConnect stream.
- “Receive Control (QzdmReceiveControl) API” on page 89 (QzdmReceiveControl) receives a control message on an OptiConnect stream.
- “Receive Request (QzdmReceiveRequest) API” on page 92 (QzdmReceiveRequest) receives a request or a message over an OptiConnect path.
- “Receive Response (QzdmReceiveResponse) API” on page 96 (QzdmReceiveResponse) receives an acknowledgement and the response data over an OptiConnect path.
- “Send Request (QzdmSendRequest) API” on page 99 (QzdmSendRequest) sends a request or a message over an OptiConnect path.
- “Send Response (QzdmSendResponse) API” on page 102 (QzdmSendResponse) sends an acknowledgement and the response data over an OptiConnect path.
- “Wait Message (QzdmWaitMessage) API” on page 105 (QzdmWaitMessage) waits for a message on an OptiConnect stream.

Top | “Communications APIs,” on page 1 | APIs by category

---

## Close Path (QzdmClosePath) API

Required Parameter Group:	
1	Request variable
<b>Input</b>	Char(*)
2	Length of request variable
<b>Input</b>	Binary(4)
3	Format name of request variable
<b>Input</b>	Char(8)
4	Error code
<b>I/O</b>	Char(*)
Library Name / Service Program: QSOC/QZDMMDTA	
Default Public Authority: *EXCLUDE	
Threadsafe: No	

The Close Path (QzdmClosePath) API is used to close an OptiConnect path. The Close Path (QzdmClosePath) API should be performed after the path is no longer needed to free the system resources associated with the path.

The system that initiated the last transaction, by using the Send Request (QzdmSendRequest) API, should be the system that closes the path after the transaction is completed with the Receive Response (QzdmReceiveResponse) API. If the system that received the request using the Receive Request (QzdmReceiveRequest) API is the system that closes the path after issuing the Send Response (QzdmSendResponse) API, then unpredictable results may occur. This is due to the Close Path (QzdmClosePath) API being able to close the path before the response is actually received by the other system that uses the Receive Response (QzdmReceiveResponse) API.



After the Close Path (QzdmClosePath) API has been issued, the other system should complete the close sequence by issuing the Receive Control (QzdmReceiveControl) API to receive the close path message from the closing system.

## Restrictions

The following restrictions apply:

- The OptiConnect QSOC subsystem must be started on both the local and remote systems prior to calling this API.
- A stream must be opened to the OptiConnect device driver on the local system by using the Open Stream (QzdmOpenStream) API prior to calling this API.
- A path must be opened to the remote system by using the Open Path (QzdmOpenPath) API prior to calling this API.

## Authority and Locks

*Service Program Authority*  
\*EXECUTE

## Required Parameter Group

### Request variable

INPUT; CHAR(\*)

The request variable structure that describes the input for the Close Path (QzdmClosePath) API.

### Length of request variable

INPUT; BINARY(4)

The length of the request variable, in bytes. The length of the request variable must be at least equal to the length of the input format, and less than or equal to the maximum length of 4KB.

### Format name of request variable

INPUT; CHAR(8)

The format of the information that is provided as input for the Close Path (QzdmClosePath) API. The format CPTH0100 is the only supported format used by this API for the request variable. See “CPTH0100 Format” for more information on the CPTH0100 format.

### Error code

I/O; CHAR(\*)

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

## CPTH0100 Format

The following table defines the information required for Format CPTH0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(16)	Stream identifier
16	10	CHAR(8)	Path identifier

## Field Descriptions

**Path identifier.** The OptiConnect path that is to be closed. This field is provided as output with the Open Path (QzdmOpenPath) API.

**Stream identifier.** The OptiConnect stream that is to be used for communications. This field is provided as output with the Open Stream (QzdmOpenStream) API.

## Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFADF0 E	The OptiConnect QSOC subsystem must be active.
CPFADF1 E	OptiConnect communication error.
CPFADF3 E	OptiConnect path not valid or closed.
CPFADF5 E	OptiConnect API internal error, function code &1, return code &2.
CPFADF6 E	Request variable not valid, reason code &1.

API introduced: V3R7

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

---

## Close Stream (QzdmCloseStream) API

Required Parameter Group:	
1	Request variable
<b>Input</b>	Char(*)
2	Length of request variable
<b>Input</b>	Binary(4)
3	Format name of request variable
<b>Input</b>	Char(8)
4	Error code
<b>I/O</b>	Char(*)
Library Name / Service Program: QSOC/QZDMMDTA	
Default Public Authority: *EXCLUDE	
Threadsafe: No	

The Close Stream (QzdmCloseStream) API is used to close an OptiConnect stream. The Close Stream (QzdmCloseStream) API should be performed after the stream is no longer needed to free the system resources associated with the stream.

## Restrictions

The following restrictions apply:

- The OptiConnect QSOC subsystem must be started on the system prior to calling this API.
- A stream must be opened to the OptiConnect device driver on the system by using the Open Stream (QzdmOpenStream) API prior to calling this API.

## Authority and Locks

Service Program Authority  
\*EXECUTE

## Required Parameter Group

### Request variable

INPUT; CHAR(\*)

The request variable structure that describes the input for the Close Stream (QzdmCloseStream) API.

### Length of request variable

INPUT; BINARY(4)

The length of the request variable, in bytes. The length of the request variable must be at least equal to the length of the input format, and less than or equal to the maximum length of 4KB.

### Format name of request variable

INPUT; CHAR(8)

The format of the information that is provided as input for the Close Stream (QzdmCloseStream) API. The CSTR0100 format is used by this API for the request variable. See "CSTR0100 Format" for more information on the CSTR0100 format.

### Error code

I/O; CHAR(\*)

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

## CSTR0100 Format

The following table defines the information required for Format CSTR0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(16)	Stream identifier

## Field Descriptions

**Stream identifier.** The OptiConnect stream that is to be closed. This field is provided as output with the Open Stream (QzdmOpenStream) API.

## Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFADF0 E	The OptiConnect QSOC subsystem must be active.
CPFADF1 E	OptiConnect communication error.
CPFADF5 E	OptiConnect API internal error, function code &1, return code &2.
CPFADF6 E	Request variable not valid, reason code &1.

API introduced: V3R7

## Open Path (QzdmOpenPath) API

Required Parameter Group:	
1	Receiver variable
<b>Output</b>	Char(*)
2	Length of receiver variable
<b>Input</b>	Binary(4)
3	Format name of receiver variable
<b>Input</b>	Char(8)
4	Request variable
<b>Input</b>	Char(*)
5	Length of request variable
<b>Input</b>	Binary(4)
6	Format name of request variable
<b>Input</b>	Char(8)
7	Error code
<b>I/O</b>	Char(*)
Library Name / Service Program: QSOC/QZDMMDTA	
Default Public Authority: *EXCLUDE	
Threadsafe: No	

The Open Path (QzdmOpenPath) API is used to open an OptiConnect path. The Open Path (QzdmOpenPath) API returns a path identifier that is then required as input for subsequent OptiConnect APIs that require a path identifier.

### Restrictions

The following restrictions apply:

- The OptiConnect QSOC subsystems must be started on both the local and remote systems prior to calling this API.
- A stream must be opened to the OptiConnect device driver on the local system by using the Open Stream (QzdmOpenStream) API prior to calling this API.
- A user profile must exist on the remote system by the same name as the user profile that is running the Open Path (QzdmOpenPath) API on the local system.  
It is the responsibility of the user to verify that the user profile name on the remote system is the same as the user profile name on the local system. The purpose of this verification is to ensure that the user's authority is the same on both systems.
- If a job description (\*JOBID) is specified in the user profile on the remote system, the job description must also reside on the remote system.
- A maximum of 256 path identifiers may be opened for a single job.

## Authority and Locks

Service Program Authority  
\*EXECUTE

## Required Parameter Group

### Receiver variable

OUTPUT; CHAR(\*)

The receiver variable that is to receive the output control information from the Open Path (QzdmOpenPath) API.

### Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable, in bytes. The length of the receiver variable must be at least equal to or greater than the length of the output format.

### Format name of receiver variable

INPUT; CHAR(8)

The format of the information that is returned from the Open Path (QzdmOpenPath) API. The OPRC0100 format is used by this API for the receiver variable. See "OPRC0100 Format" for more information on the OPRC0100 format.

### Request variable

INPUT; CHAR(\*)

The request variable structure that describes the input for the Open Path (QzdmOpenPath) API.

### Length of request variable

INPUT; BINARY(4)

The length of the request variable, in bytes. The length of the request variable must be at least equal to the length of the input format, and less than or equal to the maximum length of 4KB.

### Format name of request variable

INPUT; CHAR(8)

The format of the information that is provided as input for the Open Path (QzdmOpenPath) API. The OPRQ0100 format is used by this API for the request variable. See "OPRQ0100 Format" for more information on the OPRQ0100 format.

### Error code

I/O; CHAR(\*)

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

## OPRC0100 Format

The following table defines the information returned for Format OPRC0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(8)	Path identifier

## OPRQ0100 Format

The following table defines the information required for format OPRQ0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(16)	Stream identifier
16	10	CHAR(8)	Remote system name
24	18	CHAR(10)	Program name
34	22	CHAR(10)	Program library name

## Field Descriptions

**Path identifier.** The OptiConnect path that is to be used for communications. This field is provided as output with the Open Path (QzdmOpenPath) API. This field must then be provided as input on all subsequent OptiConnect APIs that require a path identifier.

The path identifier is associated with the stream identifier that is provided as input, as a stream-identifier and path-identifier pair. For most applications, this stream-identifier and path-identifier pair needs to be used for all subsequent OptiConnect APIs that are used to control communications on the local system.

**Remote system name.** The name of the remote system to which the OptiConnect path is being opened. This is the current system name as displayed on the Display Network Attributes (DSPNETA) display on the remote system.

**Program name.** The program name on the remote system that controls communications on the remote system. This program is called by the OptiConnect agent job (QZDMAGNT) on the remote system, and is passed a stream-identifier and path-identifier pair.

For most applications, this stream-identifier and path-identifier pair needs to be used for all subsequent OptiConnect APIs that are used to control communications on the remote system.

**Program library name.** The program library name on the remote system in which the program is contained.

**Stream identifier.** The OptiConnect stream that is to be used for communications. This field is provided as output on the Open Stream (QzdmOpenStream) API.

The stream identifier is associated with the path identifier that is provided as output, as a stream-identifier and path-identifier pair. For most applications, this stream-identifier and path-identifier pair needs to be used for all subsequent OptiConnect APIs that are used to control communications on the local system.

## Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFADF0 E	The OptiConnect QSOC subsystem must be active.
CPFADF1 E	OptiConnect communication error.
CPFADF2 E	OptiConnect path open error.
CPFADF5 E	OptiConnect API internal error, function code &1, return code &2.
CPFADF6 E	Request variable not valid, reason code &1.
CPFADF7 E	OptiConnect API open path error, function code &1, return code &2.

Message ID	Error Message Text
CPFADF8 E	Program name not found.
CPFADF9 E	Program library name not found.
CPFADFA E	User not authorized to program.
CPFADFB E	Open path rejected.
CPFADFD E	Remote system &1 not found or not valid.

API introduced: V3R7

Top | "Communications APIs," on page 1 | APIs by category

## Open Stream (QzdmOpenStream) API

Required Parameter Group:	
<b>1</b>	Receiver variable
<b>Output</b>	Char(*)
<b>2</b>	Length of receiver variable
<b>Input</b>	Binary(4)
<b>3</b>	Format name of receiver variable
<b>Input</b>	Char(8)
<b>4</b>	Error code
<b>I/O</b>	Char(*)
	Library Name / Service Program: QSOC/QZDMMDTA
	Default Public Authority: *EXCLUDE
	Threadsafe: No

The Open Stream (QzdmOpenStream) API is used to open an OptiConnect stream. The Open Stream (QzdmOpenStream) API returns a stream identifier, which is then required as input for subsequent OptiConnect APIs that require a stream identifier.

### Restrictions

The following restrictions apply:

- The OptiConnect QSOC subsystem must be started on the local system prior to calling this API.
- A maximum of 256 stream identifiers may be opened for a single job.

### Authority and Locks

*Service Program Authority*

\*EXECUTE

### Required Parameter Group

**Receiver variable**

OUTPUT; CHAR(\*)

The receiver variable that is to receive the output control information from the Open Stream (QzdmOpenStream) API.

### Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable, in bytes. The length of the receiver variable must be at least equal to or greater than the length of the output format.

### Format name of receiver variable

INPUT; CHAR(8)

The format of the information that is returned from the Open Stream (QzdmOpenStream) API. The OSTR0100 format is used by this API for the receiver variable. See "OSTR0100 Format" for more information on the OSTR0100 format.

### Error code

I/O; CHAR(\*)

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

## OSTR0100 Format

The following table defines the information returned for Format OSTR0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(16)	Stream identifier

## Field Descriptions

**Stream identifier.** The OptiConnect stream that is to be used for communications. This field is provided as output with the Open Stream (QzdmOpenStream) API. This field must then be provided as input on all subsequent OptiConnect API requests that require a stream identifier.

## Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFADF0 E	The OptiConnect QSOC subsystem must be active.
CPFADF1 E	OptiConnect communication error.
CPFADF5 E	OptiConnect API internal error, function code &1, return code &2.

API introduced: V3R7

[Top](#) | ["Communications APIs,"](#) on page 1 | [APIs by category](#)



## Receive Control (QzdmReceiveControl) API

Required Parameter Group:	
1	Receiver variable
<b>Output</b>	Char(*)
2	Length of receiver variable
<b>Input</b>	Binary(4)
3	Format name of receiver variable
<b>Input</b>	Char(8)
4	Request variable
<b>Input</b>	Char(*)
5	Length of request variable
<b>Input</b>	Binary(4)
6	Format name of request variable
<b>Input</b>	Char(8)
7	Error code
<b>I/O</b>	Char(*)
Library Name / Service Program: QSOC/QZDMMDTA	
Default Public Authority: *EXCLUDE	
Threadsafe: No	

The Receive Control (QzdmReceiveControl) API is used to receive a control message on an OptiConnect stream.

When the Close Path (QzdmClosePath) API is issued on a system to close a path, the system that is at the other end of the path must issue the Receive Control (QzdmReceiveControl) API to complete the close path sequence. If the Receive Control (QzdmReceiveControl) API is not issued, the stream identifier that is associated with the path that is being closed is not available for subsequent communications until the control message is received.

### Restrictions

The following restrictions apply:

- The OptiConnect QSOC subsystem must be started on both the local and remote systems prior to calling this API.
- A stream must be opened to the OptiConnect device driver on the local system by using the Open Stream (QzdmOpenStream) API prior to calling this API.
- A path must be opened to the remote system by using the Open Path (QzdmOpenPath) API prior to calling this API.

### Authority and Locks

*Service Program Authority*  
\*EXECUTE

## Required Parameter Group

### Receiver variable

OUTPUT; CHAR(\*)

The receiver variable that is to receive the output control information from the Receive Control (QzdmReceiveControl) API.

### Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable, in bytes. The length of the receiver variable must be at least equal to or greater than the length of the output format.

### Format name of receiver variable

INPUT; CHAR(8)

The format of the information that is returned from Receive Control(QzdmReceiveControl) API. The RCRC0100 format is used by this API for the receiver variable. See "RCRC0100 Format" for more information on the RCRC0100 format.

### Request variable

INPUT; CHAR(\*)

The request variable structure that describes the input for the Receive Control (QzdmReceiveControl) API.

### Length of request variable

INPUT; BINARY(4)

The length of the request variable, in bytes. The length of the request variable must be at least equal to the length of the input format, and less than or equal to the maximum length of 4KB.

### Format name of request variable

INPUT; CHAR(8)

The format of the information that is provided as input for the Receive Control (QzdmReceiveControl) API. The RCRQ0100 format is used by this API for the request variable. See "RCRQ0100 Format" for more information on the RCRQ0100 format.

### Error code

I/O; CHAR(\*)

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

## RCRC0100 Format

The following table defines the information returned for Format RCRC0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(1)	Control message type
1	1	CHAR(8)	Control message data

## RCRQ0100 Format

The following table defines the information required for Format RCRQ0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(16)	Stream identifier

## Field Descriptions

**Control message data.** The control message data returned for the control message type. For example, the control message data for the close path message contains the path identifier of the path that is being closed.

**Control message type.** The type of control message to be received. This field is provided as output on the Receive Control (QzdmReceiveControl) API.

The possible value follows:

1 Close path message

**Stream identifier.** The OptiConnect stream that is used for communications. This field is provided as output with the Open Stream (QzdmOpenStream) API.

## Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFADF0 E	The OptiConnect QSOC subsystem must be active.
CPFADF1 E	OptiConnect communication error.
CPFADF4 E	OptiConnect detected sequence error.
CPFADF5 E	OptiConnect API internal error, function code &1, return code &2.
CPFADF6 E	Request variable not valid, reason code &1.

API introduced: V3R7

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

## Receive Request (QzdmReceiveRequest) API

Required Parameter Group:	
1	Receiver variable
<b>Output</b>	Char(*)
2	Length of receiver variable
<b>Input</b>	Binary(4)
3	Format name of receiver variable
<b>Input</b>	Char(8)
4	Request variable
<b>Input</b>	Char(*)
5	Length of request variable
<b>Input</b>	Binary(4)
6	Format name of request variable
<b>Input</b>	Char(8)
7	Error code
<b>I/O</b>	Char(*)
Library Name / Service Program: QSOC/QZDMMDTA	
Default Public Authority: *EXECUTE	
Threadsafe: No	

The Receive Request (QzdmReceiveRequest) API is used to receive a request or a message over an OptiConnect path. A maximum of 32KB of data may be transferred in a single receive request.

### Restrictions

The following restrictions apply:

- The OptiConnect QSOC subsystem must be started on both the local and remote systems prior to calling this API.
- A stream must be opened to the OptiConnect device driver on the local system by using the Open Stream (QzdmOpenStream) API prior to calling this API.
- A path must be opened to the remote system by using the Open Path (QzdmOpenPath) API prior to calling this API.
- If the receiving system does not provide a large enough data buffer to receive all of the data, the data that will fit into the data buffer is moved, but the remaining data is truncated. The user must then increase the size of the data buffer and then retry the entire transaction.
- A maximum of 16 transactions may be in progress for a stream-identifier and path-identifier pair.

### Authority and Locks

*Service Program Authority*  
\*EXECUTE

## Required Parameter Group

### Receiver variable

OUTPUT; CHAR(\*)

The receiver variable that is to receive the output control information from the Receive Request (QzdmReceiveRequest) API.

### Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable, in bytes. The length of the receiver variable must be at least equal to or greater than the length of the output format.

### Format name of receiver variable

INPUT; CHAR(8)

The format of the information that is returned from the Receive Request (QzdmReceiveRequest) API. The RQRC0100 format is used by this API for the receiver variable. See "RQRC0100 Format" for more information on the RQRC0100 format.

### Request variable

INPUT; CHAR(\*)

The request variable structure that describes the input for the Receive Request (QzdmReceiveRequest) API.

### Length of request variable

INPUT; BINARY(4)

The length of the request variable, in bytes. The length of the request variable must be at least equal to the length of the input format, and less than or equal to the maximum length of 4KB.

### Format name of request variable

INPUT; CHAR(8)

The format of the information that is provided as input for the Receive Request (QzdmReceiveRequest) API. The RQRQ0100 format is used by this API for the request variable. See "RQRQ0100 Format" for more information on the RQRQ0100 format.

### Error code

I/O; CHAR(\*)

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

## RQRC0100 Format

The following table defines the information returned for Format RQRC0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(8)	Transaction identifier
8	8	CHAR(8)	Path identifier
16	10	BINARY(4)	Total request data length
20	14	BINARY(4)	Current output data length
24	18	BINARY(4)	Maximum response data length

## RQRQ0100 Format

The following table defines the information required for Format RQRQ0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(16)	Stream identifier
16	10	BINARY(4)	Time-out value
20	14	BINARY(4)	Offset to output descriptors
24	18	BINARY(4)	Number of output descriptors
28	1C	CHAR(4)	Reserved
These fields repeat for each output descriptor		PTR(SPP)	Data buffer pointer
		BINARY(4)	Data buffer length
		CHAR(12)	Reserved

## Field Descriptions

**Current output data length.** The total data length of the request that was moved to the user's data buffer area. If the current output data length is less than the total request data length, then this indicates that not all of the data was received. It is the responsibility of the user's application program to retry the entire transaction by using a larger data buffer size for the Receive Request (QzdmReceiveRequest) API to receive all of the data.

**Data buffer length.** The length of the data buffer that is used for receiving data.

**Data buffer pointer.** The pointer to the data buffer that is used for receiving data.

**Maximum response data length.** The maximum length that is allowed for the response data. This field is provided by the user as input on the Send Request (QzdmSendRequest) API and indicates the maximum response data length allowed for the Send Response (QzdmSendResponse) API.

**Number of output descriptors.** The number of output descriptors that are used. An output descriptor describes where the output data may be found. The output descriptor consists of a space pointer to a data buffer and the length of the data buffer. A maximum of three output descriptors may be specified.

**Offset to output descriptors.** The offset to the output descriptors.

**Path identifier.** The OptiConnect path that is to be used for communications. This field is provided as output on the Receive Request (QzdmReceiveRequest) API.

**Reserved.** A reserved space for the purpose of aligning pointer values on a 16-byte boundary. This field must be initialized to binary 0.

**Stream identifier.** The OptiConnect stream that is used for communications. This field is provided as output on the Open Stream (QzdmOpenStream) API.

**Time-out value.** A length of time, in milliseconds, to wait for the Receive Request (QzdmReceiveRequest) API to complete. If the Receive Request (QzdmReceiveRequest) API does not complete before the specified time-out value, then the exception CPFADFE is returned. The user should then re-issue the Receive Request (QzdmReceiveRequest) API and specify the same time-out value or an increased time-out value.

The Receive Request (QzdmReceiveRequest) API remains outstanding, and control is not returned to the user application until either of the following occurs:

- The request either completes successfully or unsuccessfully.
- The time-out value has been exceeded.

A value of -1 may be specified, which indicates to wait forever for the Receive Request (QzdmReceiveRequest) API to complete.

**Total request data length.** The total data length of the request that is available to be received. This field is provided as output on the Receive Request (QzdmReceiveRequest) API.

**Transaction identifier.** The specific transaction associated with this Receive Request (QzdmReceiveRequest) API. This field is provided as output on the Receive Request (QzdmReceiveRequest) API. This field must then be provided as input on the corresponding Send Response (QzdmSendResponse) API.

## Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFADF0 E	The OptiConnect QSOC subsystem must be active.
CPFADF1 E	OptiConnect communication error.
CPFADF3 E	OptiConnect path not valid or closed.
CPFADF4 E	OptiConnect detected sequence error.
CPFADF5 E	OptiConnect API internal error, function code &1, return code &2.
CPFADF6 E	Request variable not valid, reason code &1.
CPFADFE E	Time-out occurred.
CPFADFF E	Transaction was terminated.

API introduced: V3R7

[Top](#) | [“Communications APIs,”](#) on page 1 | [APIs by category](#)

## Receive Response (QzdmReceiveResponse) API

Required Parameter Group:	
1	Receiver variable
<b>Output</b>	Char(*)
2	Length of receiver variable
<b>Input</b>	Binary(4)
3	Format name of receiver variable
<b>Input</b>	Char(8)
4	Request variable
<b>Input</b>	Char(*)
5	Length of request variable
<b>Input</b>	Binary(4)
6	Format name of request variable
<b>Input</b>	Char(8)
7	Error code
<b>I/O</b>	Char(*)
Library Name / Service Program: QSOC/QZDMMDTA	
Default Public Authority: *EXECUTE	
Threadsafe: No	

The Receive Response (QzdmReceiveResponse) API is used to receive an acknowledgement and the response data over an OptiConnect path. A maximum of 32KB of data may be received in a single receive response.

The response data is received into the output buffers, which were previously defined in the output descriptors on the Send Request (QzdmSendRequest) API.

### Restrictions

The following restrictions apply:

- The OptiConnect QSOC subsystem must be started on both the local and remote systems prior to calling this API.
- A stream must be opened to the OptiConnect device driver on the local system by using the Open Stream (QzdmOpenStream) API prior to calling this API.
- A path must be opened to the remote system by using the Open Path (QzdmOpenPath) API prior to calling this API.
- If the receiving system does not provide a large enough data buffer to receive all of the data, the data that will fit into the data buffer is moved, but the remaining data is truncated. The user must then increase the size of the data buffer, and then retry the entire transaction.
- A maximum of 16 transactions may be in progress for a stream-identifier and path-identifier pair.



## Authority and Locks

Service Program Authority  
\*EXECUTE

## Required Parameter Group

### Receiver variable

OUTPUT; CHAR(\*)

The receiver variable that is to receive the output control information from the Receive Response (QzdmReceiveResponse) API.

### Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable, in bytes. The length of the receiver variable must be at least equal to or greater than the length of the output format.

### Format name of receiver variable

INPUT; CHAR(8)

The format of the information that is returned from the Receive Response (QzdmReceiveResponse) API. The RSRC0100 format is used by this API for the receiver variable. See "RSRC0100 Format" for more information on the RSRC0100 format.

### Request variable

INPUT; CHAR(\*)

The request variable structure that describes the input for the Receive Response (QzdmReceiveResponse) API.

### Length of request variable

INPUT; BINARY(4)

The length of the request variable, in bytes. The length of the request variable must be at least equal to the length of the input format, and less than or equal to the maximum length of 4KB.

### Format name of request variable

INPUT; CHAR(8)

The format of the information that is provided as input for the Receive Response (QzdmReceiveResponse) API. The RSRQ0100 format is used by this API for the request variable. See "RSRQ0100 Format" on page 98 for more information on the RSRQ0100 format.

### Error code

I/O; CHAR(\*)

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

## RSRC0100 Format

The following table defines the information returned for Format RSRC0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(4)	Acknowledgement data
4	4	BINARY(4)	Actual response data length

## RSRQ0100 Format

The following table defines the information required for Format RSRQ0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(16)	Stream identifier
16	10	CHAR(8)	Path identifier
24	18	BINARY(4)	Time-out value
28	1C	CHAR(8)	Transaction identifier

## Field Descriptions

**Acknowledgement data.** The acknowledgement data for the request. This field is provided as input on the Send Response (QzdmSendResponse) API.

**Actual response data length.** The actual length that was received for the response data. If the response data that was sent from the Send Response (QzdmSendResponse) API is larger than the buffer that was provided with the Send Request (QzdmSendRequest) API, not all of the data was received. It is the responsibility of the user's application program to retry the entire transaction by using a larger data buffer size for the Send Request (QzdmSendRequest) API to receive all of the data with the Receive Response (QzdmReceiveResponse) API.

**Path identifier.** The OptiConnect path that is used for communications. This field is provided as output on the Open Path (QzdmOpenPath) API.

**Stream identifier.** The OptiConnect stream that is used for communications. This field is provided as output on the Open Stream (QzdmOpenStream) API.

**Time-out value.** A length of time, in milliseconds, to wait for the Receive Response (QzdmReceiveResponse) API to complete. If the Receive Response (QzdmReceiveResponse) API does not complete before the specified time-out value, the exception CPFADFE is returned. The user should then re-issue the Receive Response (QzdmReceiveResponse) API and specify the same time-out value or an increased time-out value.

The Receive Response (QzdmReceiveResponse) API remains outstanding, and control is not returned to the user application until either of the following occurs:

- The request either completes successfully or unsuccessfully.
- The time-out value has been exceeded.

A value of -1 may be specified, which indicates to wait forever for the Receive Response (QzdmReceiveResponse) API to complete.

**Transaction identifier.** The specific transaction associated with this Receive Response (QzdmReceiveResponse) API. This field is provided as output on the Send Request (QzdmSendRequest) API.

## Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.

Message ID	Error Message Text
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFADF0 E	The OptiConnect QSOC subsystem must be active.
CPFADF1 E	OptiConnect communication error.
CPFADF3 E	OptiConnect path not valid or closed.
CPFADF5 E	OptiConnect API internal error, function code &1, return code &2.
CPFADF6 E	Request variable not valid, reason code &1.
CPFADFE E	Time-out occurred.
CPFADFF E	Transaction was terminated.

API introduced: V3R7

Top | "Communications APIs," on page 1 | APIs by category

## Send Request (QzdmSendRequest) API

Required Parameter Group:	
1	Receiver variable
<b>Output</b>	Char(*)
2	Length of receiver variable
<b>Input</b>	Binary(4)
3	Format name of receiver variable
<b>Input</b>	Char(8)
4	Request variable
<b>Input</b>	Char(*)
5	Length of request variable
<b>Input</b>	Binary(4)
6	Format name of request variable
<b>Input</b>	Char(8)
7	Error code
<b>I/O</b>	Char(*)
Library Name / Service Program: QSOC/QZDMMDTA	
Default Public Authority: *EXECUTE	
Threadsafe: No	

The Send Request (QzdmSendRequest) API is used to send a request or a message over an OptiConnect path. A maximum of 32KB of data may be transferred in a single send request.

### Restrictions

The following restrictions apply:

- The OptiConnect QSOC subsystem must be started on both the local and remote systems prior to calling this API.
- A stream must be opened to the OptiConnect device driver on the local system by using the Open Stream (QzdmOpenStream) API prior to calling this API.

- A path must be opened to the remote system by using the Open Path (QzdmOpenPath) API prior to calling this API.
- A maximum of 16 transactions may be in progress for a stream-identifier and path-identifier pair.

## Authority and Locks

*Service Program Authority*

\*EXECUTE

## Required Parameter Group

### Receiver variable

OUTPUT; CHAR(\*)

The receiver variable that is to receive the output control information from the Send Request (QzdmSendRequest) API.

### Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable, in bytes. The length of the receiver variable must be at least equal to or greater than the length of the output format.

### Format name of receiver variable

INPUT; CHAR(8)

The format of the information that is returned from the Send Request (QzdmSendRequest) API. The SRRC0100 format is used by this API for the receiver variable. See "SRRC0100 Format" for more information on the SRRC0100 format.

### Request variable

INPUT; CHAR(\*)

The request variable structure that describes the input for the Send Request (QzdmSendRequest) API.

### Length of request variable

INPUT; BINARY(4)

The length of the request variable, in bytes. The length of the request variable must be at least equal to the length of the input format, and less than or equal to the maximum length of 4KB.

### Format name of request variable

INPUT; CHAR(8)

The format of the information that is provided as input for the Send Request (QzdmSendRequest) API. The SRRQ0100 format is used by this API for the request variable. See "SRRQ0100 Format" on page 101 for more information on the SRRQ0100 format.

### Error code

I/O; CHAR(\*)

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

## SRRC0100 Format

The following table defines the information returned for Format SRRC0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(8)	Transaction identifier

## SRRQ0100 Format

The following table defines the information required for Format SRRQ0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(16)	Stream identifier
16	10	CHAR(8)	Path identifier
24	18	BINARY(4)	Maximum response data length
28	1C	BINARY(4)	Offset to input descriptors
32	20	BINARY(4)	Number of input descriptors
36	24	BINARY(4)	Offset to output descriptors
40	28	BINARY(4)	Number of output descriptors
44	2C	CHAR(4)	Reserved
These fields repeat for each input descriptor		PTR(SPP)	Data buffer pointer
		BINARY(4)	Data buffer length
		CHAR(12)	Reserved
These fields repeat for each output descriptor		PTR(SPP)	Data buffer pointer
		BINARY(4)	Data buffer length
		CHAR(12)	Reserved

## Field Descriptions

**Data buffer length.** The length of the data buffer that is used for the input or output data.

**Data buffer pointer.** The pointer to the input data buffer that is used for input or output data.

**Maximum response data length.** The maximum length that is allowed for the response data. This field is provided as output on the Receive Request (QzdmReceiveRequest) API and indicates the maximum response data length allowed for the Send Response (QzdmSendResponse) API. If the response data that is sent from the Send Response (QzdmSendResponse) API is larger than the buffer that is provided with the Send Request (QzdmSendRequest) API, not all of the data is received. It is the responsibility of the user's application program to retry the entire transaction by using a larger data buffer size for the Send Request (QzdmSendRequest) API to receive all of the data with the Receive Response (QzdmReceiveResponse) API.

**Number of output descriptors.** The number of output descriptors that are used. An output descriptor describes where the output data that is to be received from the remote system may be found. The output descriptor consists of a space pointer to a data buffer and the length of the data buffer. A maximum of three output descriptors may be specified. The total length of the output buffers must be equal to the maximum response data length that is specified.

**Number of input descriptors.** The number of input descriptors that are used. An input descriptor describes where the input data that is to be sent to the remote system may be found. The input descriptor consists of a space pointer to a data buffer and the length of the data buffer. A maximum of three input descriptors may be specified.

**Offset to output descriptors.** The offset to the output descriptors.

**Offset to input descriptors.** The offset to the input descriptors.

**Path identifier.** The OptiConnect path that is used for communications. This field is provided as output on the Open Path (QzdmOpenPath) API.

**Reserved.** A reserved space for the purpose of aligning pointer values on a 16-byte boundary. This field must be initialized to binary 0.

**Stream identifier.** The OptiConnect stream that is used for communications. This field is provided as output on the Open Stream (QzdmOpenStream) API.

**Transaction identifier.** The specific transaction associated with this Send Request. This field is provided as output on the Send Request (QzdmSendRequest) API. This field must then be provided as input on the corresponding Receive Response (QzdmReceiveResponse) API.

## Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFADF0 E	The OptiConnect QSOC subsystem must be active.
CPFADF1 E	OptiConnect communication error.
CPFADF3 E	OptiConnect path not valid or closed.
CPFADF5 E	OptiConnect API internal error, function code &1, return code &2.
CPFADF6 E	Request variable not valid, reason code &1.

API introduced: V3R7

[Top](#) | ["Communications APIs," on page 1](#) | [APIs by category](#)

---

## Send Response (QzdmSendResponse) API

Required Parameter Group:	
1	Request variable
<b>Input</b>	Char(*)
2	Length of request variable
<b>Input</b>	Binary(4)
3	Format name of request variable
<b>Input</b>	Char(8)
4	Error code
<b>I/O</b>	Char(*)
Library Name / Service Program: QSOC/QZDMMDTA	
Default Public Authority: *EXECUTE	
Threadsafe: No	

The Send Response (QzdmSendResponse) API is used to send an acknowledgement and the response data over an OptiConnect path. A maximum of 32KB of data may be transferred in a single send response.

## Restrictions

The following restrictions apply:

- The OptiConnect QSOC subsystem must be started on both the local and remote systems prior to calling this API.
- A stream must be opened to the OptiConnect device driver on the local system by using the Open Stream (QzdmOpenStream) API prior to calling this API.
- A path must be opened to the remote system by using the Open Path (QzdmOpenPath) API prior to calling this API.
- If the receiving system does not provide a large enough data buffer to receive all of the data, the data that will fit into the data buffer is moved, but the remaining data is truncated. The user must increase the size of the data buffer and then retry the entire transaction.
- A maximum of 16 transactions may be in progress for a stream-identifier and path-identifier pair.

## Authority and Locks

*Service Program Authority*

\*EXECUTE

## Required Parameter Group

### Request variable

INPUT; CHAR(\*)

The request variable structure that describes the input for the Send Response (QzdmSendResponse) API.

### Length of request variable

INPUT; BINARY(4)

The length of the request variable, in bytes. The length of the request variable must be at least equal to the length of the input format, and less than or equal to the maximum length of 4KB.

### Format name of request variable

INPUT; CHAR(8)

The format of the information that is provided as input for the Send Response (QzdmSendResponse) API. The SRSP0100 format is used by this API for the request variable. See “SRSP0100 Format” for more information on the SRSP0100 format.

### Error code

I/O; CHAR(\*)

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

## SRSP0100 Format

The following table defines the information required for Format SRSP0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(16)	Stream identifier
16	10	CHAR(8)	Transaction identifier
24	18	BINARY(4)	Actual response data length

Offset		Type	Field
Dec	Hex		
28	1C	CHAR(4)	Acknowledgement data
32	20	BINARY(4)	Offset to input descriptors
36	24	BINARY(4)	Number of input descriptors
40	28	CHAR(8)	Reserved
These fields repeat for each input descriptor		PTR(SPP)	Data buffer pointer
		BINARY(4)	Data buffer length
		CHAR(12)	Reserved

## Field Descriptions

**Acknowledgement data.** The acknowledgement data for the request. This field is provided as output on the Receive Response (QzdmReceiveResponse) API and indicates the acknowledgement data.

**Actual response data length.** The actual length that is sent for the response data. If the response data that is sent is larger than the buffer that is provided on the Send Request (QzdmSendRequest) API, not all of the data is sent. It is the responsibility of the user's application program to retry the entire transaction by using a larger data buffer size for the Send Request (QzdmSendRequest) API to receive all of the data with the Receive Response (QzdmReceiveResponse) API.

**Data buffer length.** The length of the data buffer that is used for sending data.

**Data buffer pointer.** The pointer to the data buffer that is used for sending data.

**Number of input descriptors.** The number of input descriptors that are used. An input descriptor describes where the input data may be found. The input descriptor consists of a space pointer to a data buffer and the length of the data buffer. A maximum of three input descriptors may be specified.

**Offset to input descriptors.** The offset to the input descriptors.

**Reserved.** A reserved space for the purpose of aligning pointer values on a 16-byte boundary. This field must be initialized to binary 0.

**Stream identifier.** The OptiConnect stream that is used for communications. This field is provided as output on the Open Stream (QzdmOpenStream) API.

**Transaction identifier.** The specific transaction associated with this Send Response (QzdmSendResponse) API. This field is provided as output on the Receive Request (QzdmReceiveRequest) API.

## Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFADF0 E	The OptiConnect QSOC subsystem must be active.
CPFADF1 E	OptiConnect communication error.
CPFADF3 E	OptiConnect path not valid or closed.
CPFADF4 E	OptiConnect detected sequence error.



Message ID	Error Message Text
CPFADF5 E	OptiConnect API internal error, function code &1, return code &2.
CPFADF6 E	Request variable not valid, reason code &1.
CPFADFF E	Transaction was terminated.

API introduced: V3R7

Top | "Communications APIs," on page 1 | APIs by category

## Wait Message (QzdmWaitMessage) API

Required Parameter Group:	
1	Receiver variable
<b>Output</b>	Char(*)
2	Length of receiver variable
<b>Input</b>	Binary(4)
3	Format name of receiver variable
<b>Input</b>	Char(8)
4	Request variable
<b>Input</b>	Char(*)
5	Length of request variable
<b>Input</b>	Binary(4)
6	Format name of request variable
<b>Input</b>	Char(8)
7	Error code
<b>I/O</b>	Char(*)
Library Name / Service Program: QSOC/QZDMMDTA	
Default Public Authority: *EXECUTE	
Threadsafe: No	

The Wait Message (QzdmWaitMessage) API is used to wait for a message on an OptiConnect stream. The message may be a request message, a response message, or a control message.

### Restrictions

The following restrictions apply:

- The OptiConnect QSOC subsystem must be started on both the local and remote systems prior to calling this API.
- A stream must be opened to the OptiConnect device driver on the local system by using the Open Stream (QzdmOpenStream) API prior to calling this API.
- A path must be opened to the remote system by using the Open Path (QzdmOpenPath) API prior to calling this API.

## Authority and Locks

Service Program Authority  
\*EXECUTE

## Required Parameter Group

### Receiver variable

OUTPUT; CHAR(\*)

The receiver variable that is to receive the output control information from the Wait Message (QzdmWaitMessage) API.

### Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable, in bytes. The length of the receiver variable must be at least equal to or greater than the length of the output format.

### Format name of receiver variable

INPUT; CHAR(8)

The format of the information that is returned from the Wait Message (QzdmWaitMessage) API. The WMRC0100 format is used by this API for the receiver variable. See “WMRC0100 Format” for more information on the WMRC0100 format.

### Request variable

INPUT; CHAR(\*)

The request variable structure that describes the input for the Wait Message (QzdmWaitMessage) API.

### Length of request variable

INPUT; BINARY(4)

The length of the request variable, in bytes. The length of the request variable must be at least equal to the length of the input format, and less than or equal to the maximum length of 4KB.

### Format name of request variable

INPUT; CHAR(8)

The format of the information that is provided as input for the Wait Message (QzdmWaitMessage) API. The WMRQ0100 format is used by this API for the request variable. See “WMRQ0100 Format” for more information on the WMRQ0100 format.

### Error code

I/O; CHAR(\*)

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

## WMRC0100 Format

The following table defines the information returned for Format WMRC0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(1)	Message type

## WMRQ0100 Format

The following table defines the information required for Format WMRQ0100.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(16)	Stream identifier
16	10	BINARY(4)	Time-out value

## Field Descriptions

**Message type.** The type of message that is received. This field is provided as output on the Wait Message (QzdmWaitMessage) API.

Possible values follow:

1	Request message
2	Response message
3	Control message

**Stream identifier.** The OptiConnect stream that is used for communications. This field is provided as output on the Open Stream (QzdmOpenStream) API.

**Time-out value.** A length of time, in milliseconds, to wait for the Wait Message (QzdmWaitMessage) API to complete. If the Wait Message (QzdmWaitMessage) API does not complete before the specified time-out value, the exception CPFADFE is returned. The user should then re-issue the Wait Message (QzdmWaitMessage) API and specify the same time-out value or an increased time-out value.

The Wait Message (QzdmWaitMessage) API remains outstanding, and control is not returned to the user application until either of the following occurs:

- The request either completes successfully or unsuccessfully.
- The time-out value has been exceeded.

A value of -1 may be specified, which indicates to wait forever for the Wait Message (QzdmWaitMessage) API to complete.

## Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1D E	Length specified in parameter &1 not valid.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
CPFADF0 E	The OptiConnect QSOC subsystem must be active.
CPFADF1 E	OptiConnect communication error.
CPFADF5 E	OptiConnect API internal error, function code &1, return code &2.
CPFADF6 E	Request variable not valid, reason code &1.
CPFADFE E	Time-out occurred.

API introduced: V3R7

---

## TCP/IP Management

The TCP/IP Management APIs allow you to retrieve information about your TCP/IP setup and status, and change certain system values related to TCP/IP.

The TCP/IP Management APIs are:

- “Change Connection Attribute (QTOCCCNA) API” on page 109 (QTOCCCNA) can change the attribute of a socket or connection directly.
- “List Neighbor Cache Table (QtocLstNeighborTbl) API” on page 111 (QtocLstNeighborTbl) returns a list of all entries in the IPv6 Neighbor Cache table for a specified line or for all lines.
- “List Network Connections (QtocLstNetCnn) API” on page 116 (QtocLstNetCnn) returns a non-detailed list of all the network connections for a specified net connection type or a list of the subset of network connections for a specified net connection.
- “List Network Interfaces (QtocLstNetIfc) API” on page 126 (QtocLstNetIfc) returns a detailed list of all logical TCP/IP interfaces.
- “List Network Routes (QtocLstNetRte) API” on page 139 (QtocLstNetRte) returns a detailed list of all routes.
- “List Physical Interface ARP Table (QtocLstPhyIfcARPTbl) API” on page 150 (QtocLstPhyIfcARPTbl) returns a list of all entries in the Address Resolution Protocol (ARP) table for the specified time.
- “List Physical Interface Data (QtocLstPhyIfcDta) API” on page 154 (QtocLstPhyIfcDta) returns a list of physical interfaces and detailed information about TCP/IP-related data for each of the listed physical interfaces.
- “List PPP Connection Profiles (QtocLstPPPCnnPrf) API” on page 167 (QtocLstPPPCnnPrf) returns a list of PPP connection profiles with some basic information about each profile.
- “List TCP/IP Point-to-Point Jobs (QTOCLPPJ) API” on page 172 (QTOCLPPJ) returns information about each connection job currently associated with the specified point-to-point connection profile.
- “Remove ARP Table Entry (QtocRmvARPTblE) API” on page 175 (QtocRmvARPTblE) removes one or all dynamic entries from the ARP table for the specified line.
- “Retrieve Network Connection Data (QtocRtvNetCnnDta) API” on page 177 (QtocRtvNetCnnDta) retrieves the details of any specified connection-including jobs using the connection.
- “Retrieve PPP Connection Profiles (QtocRtvPPPCnnPrf) API” on page 196 (QtocRtvPPPCnnPrf) retrieves the details of a specific PPP connection job profile.
- “Retrieve TCP/IP Attributes (QtocRtvTCPA) API” on page 212 (QtocRtvTCPA) retrieves TCP/IP attributes.
- “Update DNS API (QTOBUPDT)” on page 228 (QTOBUPDT) allows the caller to send one or more update instructions to an iSeries dynamic DNS (Domain Name System) server.

See Resource Reservation Setup Protocol APIs for information on APIs that perform your integrated services reservation.

The TCP/IP Management exit programs are:

- FTP client request validation exit point allows you to restrict operations performed by FTP users.
- FTP server request validation exit point allows you to restrict operations performed by FTP users.
- FTP server logon exit point allows you to control the authentication of users to a TCP/IP application server.
- REXEC server command processing selection (QIBM\_QTMX\_SVR\_SELECT) exit point allows you to specify which command processor the REXEC server uses for interpreting and running commands.
- TCP/IP request validation (QIBM\_QTMX\_SERVER\_REQ) exit point allows you to restrict operations on the REXEC server.
- TCP/IP server logon (QIBM\_QTMX\_SVR\_LOGON) exit point allows you to control the authentication of users and setting up user environments for the REXEC server.

- Telnet device initialization exit program allows you to associate your custom exit program with exit points on the iSeries Telnet server.
- Telnet device termination exit program allows you to log session termination information.
- TFTP request validation (QIBM\_QTOD\_SERVER\_REQ) exit point allows you to restrict operations on the TFTP server.
- **»** “Trace Exit Program for Trace TCP Application command” on page 234 indicates if the trace should stop or continue running.**«**
- **»** “Exit Program for Watch for Trace Event” on page 237 is called while using commands to watch for specific events, such as messages being sent to a particular queue.**«**

Top | “Communications APIs,” on page 1 | APIs by category

---

## Change Connection Attribute (QTOCCCNA) API

Required Parameter Group:	
1	Change information
<b>Input</b>	Char(*)
2	Length of change information
<b>Input</b>	Binary(4)
3	Change information format
<b>Input</b>	Char(8)
4	Error code
<b>I/O</b>	Char(*)
Threadsafe: Yes	

The **Change Connection Attribute (QTOCCCNA)** API can change the attribute of a socket or connection directly. A valid socket descriptor is not required. Instead, the socket or connection to be changed is identified by specifying the associated port and IP address information.

The SO\_DEBUG socket option is the only attribute that can be changed.

## Authorities and Locks

*Default public authority*  
\*EXCLUDE.

## Required Parameter Group

### Change information

INPUT; CHAR(\*)

The socket or connection that is changed.

### Change information format

INPUT; CHAR(8)

The format of the change information input data. The possible values are:

TCPA0001

Change the connection attribute of a connection. The connection is identified by specifying the local and remote values for the IP address and port number. See “TCPA0001 Format” on page 110 below.

UDPA0001

Change the connection attribute of a socket. The socket is identified by specifying its local IP address and local port number. See “UDPA0001 Format” below.

### Length of change information

INPUT; BINARY(4)

The total length in bytes of the change information input variable.

### Error code

I/O; CHAR(\*)

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

## TCPA0001 Format

Use this format when changing a connection. For detailed descriptions of the fields in this table, see “Field Descriptions.”

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Attribute to change
4	4	BINARY(4)	Attribute value
8	8	BINARY(4)	Local IP address
12	C	BINARY(4)	Local port number
16	10	BINARY(4)	Remote IP address
20	14	BINARY(4)	Remote port number
24	18		

## UDPA0001 Format

Use this format when changing a socket. For detailed descriptions of the fields in this table, see “Field Descriptions.”

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Attribute to change
4	4	BINARY(4)	Attribute value
8	8	BINARY(4)	Local IP address
12	C	BINARY(4)	Local port number
16	10		

## Field Descriptions

**Attribute to change.** The possible value is:

1

Change the debug attribute (SO\_DEBUG socket option) of the connection.

**Attribute value.** Possible values are:

0	The debug attribute is not set.
1	The debug attribute is set.

**Local IP address.** The local internet address used by the connection in binary form.

**Local port number.** The local system port number used by the connection.

**Remote IP address.** The remote internet address used by the connection in binary form.

**Remote port number.** The remote system port number used by the connection.

## Error Messages

Message ID	Error Message Text
CPF3C17 E	Error occurred with input data parameter.
CPF3C21 E	Format name &1 is not valid.
CPF3C1 E	Required parameter &1 omitted.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
TCP2670 E	Not able to complete request. TCP/IP services are not available.
TCP3B03 E	Connection &1:&2, &3:&4, not found.
TCP3B04 E	Socket &1:&2, &3:&4, not found.
TCP9999 E	Internal system error in program &1.

API introduced: V5R1

[Top](#) | ["Communications APIs," on page 1](#) | [APIs by category](#)

---

## List Neighbor Cache Table (QtocLstNeighborTbl) API

Required Parameter Group:	
1	Qualified user space name
<b>Input</b>	Char(20)
2	Format name
<b>Input</b>	Char(8)
3	Line name
<b>Input</b>	Char(10)
4	Error Code
<b>I/O</b>	Char(*)
Service Program: QTOCNETSTS Threadsafes: Yes	

The List Neighbor Cache Table (QtocLstNeighborTbl) API returns a list of all entries in the IPv6 Neighbor Cache table for a specified line or for all lines.

TCP/IP must be active on this system; otherwise, error message TCP84C0 will be issued.

## Authorities and Locks

*User Space Library Authority*  
\*EXECUTE

User Space Authority

\*CHANGE

User Space Lock

\*SHRNUP

## Required Parameter Group

### Qualified user space name

INPUT; CHAR(20)

The user space that receives the information, and the library in which it is located. The first 10 characters contain the user space name, and the second 10 characters contain the library name. You can use these special values for the library name:

\*CURLIB            The job's current library.

\*LIBL              The library list.

### Format name

INPUT; CHAR(8)

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

NNCT0100           List of Neighbor Cache table entries for a specified line. Refer to "NNCT0100 Format" on page 113 for details on the format.

### Line name

INPUT; CHAR(10)

The name of the IPv6 enabled physical interface for which to retrieve Neighbor Cache table entries. The following special value may be used:

\*ALL                Request all Neighbor Cache entries for all IPv6 enabled lines in the system.

### Error Code

I/O; CHAR(\*)

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

## Format of Neighbor Cache Table Lists

To request a list of Neighbor Cache table entries for a line, use format NNCT0100.

The Neighbor Cache table list consists of:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:
  - NNCT0100 format.

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections.

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API Examples.



## Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name specified
28	1C	CHAR(10)	Line name specified

## Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name
10	A	CHAR(10)	User space library name used
20	14	CHAR(10)	Line name used

## NNCT0100 Format

The following information about an entry in the Neighbor Cache table is returned for the NNCT0100 format. For detailed descriptions of the fields in the table, see “Field Descriptions” on page 114.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(45)	Internet IPv6 address
45	2D	CHAR(3)	Reserved
48	30	CHAR(16)	Internet IPv6 address binary
64	40	CHAR(17)	Link layer address
81	51	CHAR(7)	Reserved
88	58	BINARY(8)	Link layer address binary
96	60	CHAR(10)	Line name
106	6A	CHAR(2)	Reserved
108	6C	BINARY(4)	Reachability state
112	70	CHAR(8)	Reachability state change - date
120	78	CHAR(9)	Reachability state change - time
129	81	CHAR(3)	Reserved
132	84	BINARY(4)	Reachability state error information
136	88	BINARY(4)	Time in reachable state
140	8C	BINARY(4)	Is router
144	90	BINARY(4)	Number of unicast neighbor solicitation packets sent
148	94	BINARY(4)	Number of multicast neighbor solicitation packets sent
152	98	BINARY(4)	Delay first probe time
156	9C	BINARY(4)	Max unicast solicits
160	A0	BINARY(4)	Max multicast solicits

## Field Descriptions

**Delay first probe time.** The current value of the configured stack attribute named Neighbor solicitation delay first probe time. This attribute controls how long a Neighbor Cache entry will stay in the DELAY state before the stack will send another Neighbor Solicitation and move this Neighbor Cache entry's Reachability state to PROBE if reachability still has not been confirmed. Valid values range from 3 through 10 seconds.

**Internet IPv6 address.** The IPv6 address of the neighbor in IPv6 address format notation. This field is NULL padded.

**Internet IPv6 address binary.** The binary representation of the neighbor's IPv6 address. Even though this field is defined as a character field, binary data will be returned in it.

**Is router.** Whether this neighbor is a router. Possible values are:

0	No, this neighbor is not a router.
1	Yes, this neighbor is a router.

**Line name.** The name of the communications line description that identifies the physical interface which is directly connected to this neighbor.

**Link layer address.** The MAC address of the neighbor's network interface. Format: XX:XX:XX:XX:XX:XX, where each 'X' is a hexadecimal digit.

**Link layer address binary.** The binary representation of the neighbor's six byte link layer address.

**Max multicast solicits.** The current value of the configured Neighbor solicitation max multicast solicits stack attribute. This attribute controls the maximum number of multicast Neighbor Solicitations which will be sent out when the system is performing link-layer address resolution for another host (neighbor). If no Neighbor Advertisement is received after the maximum number of Neighbor Solicitations have been sent out, address resolution has failed, and an ICMPv6 error message will be returned to the application. Valid values range from 1 through 5 transmissions.

**Max unicast solicits.** The current value of the configured Neighbor solicitation max unicast solicits stack attribute. This attribute controls the maximum number of unicast Neighbor Solicitations which will be sent out when the system is performing link-layer address resolution for another host with unicast Neighbor Solicitations. Multicast is the normal way to perform Neighbor Discovery, but unicast Neighbor Solicitations will be used if the local physical interface is not multicast-capable. If no Neighbor Advertisement is received after the maximum number of Neighbor Solicitations have been sent out, address resolution has failed, and an ICMPv6 error message will be returned to the application. Valid values range from 1 through 5 transmissions.

**Number of multicast neighbor solicitation packets sent.** The total number of multicast Neighbor Solicitations which have been sent from the local system to this neighbor.

**Number of unicast neighbor solicitation packets sent.** The total number of unicast Neighbor Solicitations which have been sent from the local system to this neighbor.

**Reserved.** An ignored field.

**Reachability state.** The reachability state of this neighbor cache entry. Possible values are:

-1	ERROR - An error has occurred while verifying the reachability of this neighbor. Use the returned Reachability state error information field value for more information about this error.
----	---

- 1 INCOMPLETE - Address resolution is being performed on the entry. Specifically, a Neighbor Solicitation has been sent to the solicited-node multicast address of the target, but the corresponding Neighbor Advertisement has not yet been received.
- 2 REACHABLE - Positive confirmation was received that the forward path to the neighbor was functioning properly. While REACHABLE, no special action takes place as packets are sent.
- 3 STALE - The STALE state is entered upon receiving an unsolicited Neighbor Discovery message that updates the cached link-layer address. Receipt of such a message does not confirm reachability, and entering the STALE state insures reachability is verified quickly if the entry is actually being used. However, reachability is not actually verified until the entry is actually used. While STALE, no action takes place until a packet is sent.
- 4 DELAY - This neighbor is assumed to be reachable, and the system is now trying to verify reachability.
- 5 PROBE - A reachability confirmation is actively being sought by retransmitting Neighbor Solicitations every "Retransmit interval" seconds until a reachability confirmation is received.
- 6 DELETING - The TCP/IP stack is currently in the process of deleting this neighbor entry from the Neighbor Cache.

**Reachability state change - date.** The date of the last change of this neighbor's Reachability state. The format of the characters in this field is "YYYYMMDD".

The meaning of those characters is as follows:

YYYY	Year
MM	Month
DD	Day

**Reachability state change - time.** The time of the last change of this neighbor's Reachability state. The format of the characters in this field is "HHMMSSmmm".

The meaning of those characters is as follows:

HH	Hours
MM	Minutes
SS	Seconds
mmm	Milliseconds

**Reachability state error information.** The error code for this Neighbor Cache entry when the Reachability state is ERROR. This value is only useful when the Reachability state field value is ERROR. Possible values are:

- 0 No error.
- 1 Unknown. An unknown error has occurred.

**Reserved.** An ignored field.

**Time in reachable state.** The length of time, in seconds, that this neighbor has been in the Reachable state. The following special value may be returned:

- 1 This neighbor currently is not in the Reachable state.

## Error Messages

Message ID	Error Message Text
TCP84C0 E	TCP/IP stack not active.
TCP84C3 E	The specified line name does not exist.
TCP84C5 E	API error providing TCP/IP Network Status information.
TCP84C6 E	Internal operations error - RESULT &1 CC &2 RC &3 ERRNO &4.

<b>Message ID</b>	<b>Error Message Text</b>
TCP84C9 I	Information returned incomplete.
TCP84CB E	Specified line &1 not configured for IPv6.
TCP84CC E	Specified line &1 does not support Neighbor Discovery for IPv6.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R2

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

---

## List Network Connections (QtocLstNetCnn) API

Required Parameter Group:	
<b>1</b>	Qualified user space name
<b>Input</b>	Char(20)
<b>2</b>	Format name
<b>Input</b>	Char(8)
<b>3</b>	Connection list qualifier
<b>Input</b>	Char(*)
<b>4</b>	Connection list qualifier size
<b>Input</b>	Binary(4)
<b>5</b>	Connection list qualifier format
<b>Input</b>	Char(8)
<b>6</b>	Error Code
<b>I/O</b>	Char(*)
Service Program: QTOCNETSTS	
Threadsafe: Yes	

The List Network Connections (QtocLstNetCnn) API returns a non-detailed list of all network connections, or a subset of all network connections for a specified network connection type. With each call to this API you can request IPv4 or IPv6 connections, but not both at the same time.

TCP/IP must be active on this system; otherwise error message TCP84C0 will be issued.

## Authorities and Locks

*User Space Library Authority*  
\*EXECUTE

*User Space Authority*  
\*CHANGE

*User Space Lock*  
\*SHRNUP

## Required Parameter Group

**Qualified user space name**  
INPUT; CHAR(20)

The user space that is to receive the created list. The first 10 characters contain the user space name; the second 10 characters contain the name of the library in which the user space is located. You can use these special values for the library name:

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

**Format name**  
INPUT; CHAR(8)

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

NCNN0100            Non-detailed list of selected TCP/IPv4 local system connections. Refer to "NCNN0100 Format" on page 122 for details on the format.  
NCNN0200            Non-detailed list of selected TCP/IPv6 local system connections. Refer to "NCNN0200 Format" on page 124 for details on the format.

**Connection list qualifier**  
INPUT; CHAR(\*)

A restriction on the network connections to be listed.

**Connection list qualifier size**  
INPUT; BINARY(4)

The size in bytes of the connection list qualifier parameter.

**Connection list qualifier format**  
INPUT; CHAR(8)

The format of the connection list qualifier parameter. The format name supported is:

NCLQ0100            IPv4 connection list qualifier. Refer to "NCLQ0100 Format" on page 118 for details on the format.  
NCLQ0200            IPv6 connection list qualifier. Refer to "NCLQ0200 Format" on page 120 for details on the format.

**Error code**  
I/O; CHAR(\*)

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

## Format of Connection Status Lists

To request a non-detailed list of local system connections, use format NCNN0100.

The connection description list consists of:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:
  - NCNN0100 format, or
  - NCNN0200 format

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections.

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API Examples.

## Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name specified
28	1C	CHAR(*)	Connection list qualifier specified
		BINARY(4)	Connection list qualifier size specified
		CHAR(8)	Connection list qualifier format specified

## Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name used
10	A	CHAR(10)	User space library name used

## NCLQ0100 Format

The following table shows the format of the IPv4 connection list qualifier input parameter, named the NCLQ0100 format. For detailed descriptions of the fields in the table, see “Field Descriptions” on page 119.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Net connection type
10	A	CHAR(10)	List request type
20	14	CHAR(12)	Reserved
32	20	BINARY(4)	Local internet address lower value
36	24	BINARY(4)	Local internet address upper value
40	28	BINARY(4)	Local port lower value
44	2C	BINARY(4)	Local port upper value

Offset		Type	Field
Dec	Hex		
48	30	BINARY(4)	Remote internet address lower value
52	34	BINARY(4)	Remote internet address upper value
56	38	BINARY(4)	Remote port lower value
60	3C	BINARY(4)	Remote port upper value

## Field Descriptions

**List request type.** The local internet address range, local port range, remote internet address range, and remote port range for which information is requested. Possible values are:

*\*ALL* All objects returned.  
*\*SUBSET* Restrict the objects returned in the list to a specified subset.

**Local internet address lower value.** The lower value of the local system internet address range, in dotted decimal format, requested for subsetting the list. The following is a special value:

0 Request all local internet addresses.

**Local internet address upper value.** The upper value of the local system internet address range, in dotted decimal format, requested for subsetting the list. The following is a special value:

0 Request only one local internet address specified by the local internet address lower value.

**Local port lower value.** The lower value of the local system port range requested for subsetting the list. Valid values range from 1 to 65535. The following is a special value:

0 Request all local ports.

**Local port upper value.** The upper value of the local system port range requested for subsetting the list. Valid values range from 1 to 65535. The following is a special value:

0 Request only one local port specified in local port lower value.

**Net connection type.** The type of connection or socket. Possible values are:

*\*ALL* All connection types  
*\*TCP* A transmission control protocol (TCP) connection or socket.  
*\*UDP* A User Datagram Protocol (UDP) socket.  
*\*IPI* An Internet Protocol (IP) over Internetwork Packet Exchange (IPX) connection or socket.

**Note:** As of V5R2, IP over IPX is no longer supported.

*\*IPS* An Internet Protocol (IP) over SNA connection or socket.

**Remote internet address lower value.** The lower value of the remote system internet address range, in dotted decimal format, requested for subsetting the list. The following is a special value:

0 Request all remote internet addresses.

**Remote internet address upper value.** The upper value of the remote system internet address range, in dotted decimal format, requested for subsetting the list. The following is a special value:

0 Request only one remote internet address specified by the remote internet address lower value.

**Remote port lower value.** The lower value of the remote system port range requested for subsetting the list. Valid values range from 1 to 65535. The following is a special value:

0 Request all remote ports.

**Remote port upper value.** The upper value of the remote system port range requested for subsetting the list. Valid values range from 1 to 65535. The following is a special value:

0 Request only one remote port specified in remote port lower value.

**Reserved.** A reserved field. It must be x'00'.

## NCLQ0200 Format

The following table shows the format of the IPv6 connection list qualifier input parameter, named the NCLQ0200 format. For detailed descriptions of the fields in the table, see "Field Descriptions."

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Net connection type
10	A	CHAR(10)	List request type
20	14	CHAR(12)	Reserved
32	20	CHAR(16)	Local internet IPv6 address lower value
48	30	CHAR(16)	Local internet IPv6 address upper value
64	40	BINARY(4)	Local port lower value
68	44	BINARY(4)	Local port upper value
72	48	CHAR(16)	Remote internet IPv6 address lower value
88	58	CHAR(16)	Remote internet IPv6 address upper value
104	68	BINARY(4)	Remote port lower value
108	6C	BINARY(4)	Remote port upper value

## Field Descriptions

**List request type.** The local internet address range, local port range, remote internet address range, and remote port range for which information is requested.

Possible values are:

\*ALL All objects returned.  
 \*SUBSET Restrict the objects returned in the list to a specified subset.

**Local internet IPv6 address lower value.** The lower value of the local system internet address range, in IPv6 address format, requested for subsetting the list. Even though this field is defined as a character field, it must be stored in binary. It is recommended that you use the Sockets in6\_addr structure.

The following is a special value:



0 Request all local internet IPv6 addresses. Specify this value by filling the whole field with binary NULLs (x'000000...').

**Local internet IPv6 address upper value.** The upper value of the local system internet address range, in IPv6 address format, requested for subsetting the list. Even though this field is defined as a character field, it must be stored in binary. It is recommended that you use the Sockets `in6_addr` structure.

The following is a special value:

0 Request only one local internet IPv6 address specified by the local internet IPv6 address lower value. Specify this value by filling the whole field with binary NULLs (x'000000...').

**Local port lower value.** The lower value of the local system port range requested for subsetting the list. Valid values range from 1 to 65535.

The following is a special value:

0 Request all local ports.

**Local port upper value.** The upper value of the local system port range requested for subsetting the list. Valid values range from 1 to 65535.

The following is a special value:

0 Request only one local port specified in local port lower value.

**Net connection type.** The type of connection or socket.

Possible values are:

\*ALL All connection types  
\*TCP A transmission control protocol (TCP) connection or socket.  
\*UDP A User Datagram Protocol (UDP) socket.

**Remote internet IPv6 address lower value.** The lower value of the remote system internet IPv6 address range, in IPv6 address format, requested for subsetting the list. Even though this field is defined as a character field, it must be stored in binary. It is recommended that you use the Sockets `in6_addr` structure.

The following is a special value:

0 Request all remote internet IPv6 addresses. Specify this value by filling the whole field with binary NULLs (x'000000...').

**Remote internet IPv6 address upper value.** The upper value of the remote system internet IPv6 address range, in IPv6 address format, requested for subsetting the list. Even though this field is defined as a character field, it must be stored in binary. It is recommended that you use the Sockets `in6_addr` structure.

The following is a special value:

0 Request only one remote internet IPv6 address specified by the remote internet IPv6 address lower value. Specify this value by filling the whole field with binary NULLs (x'000000...').

**Remote port lower value.** The lower value of the remote system port range requested for subsetting the list. Valid values range from 1 to 65535.

The following is a special value:

0 Request all remote ports.

**Remote port upper value.** The upper value of the remote system port range requested for subsetting the list. Valid values range from 1 to 65535.

The following is a special value:

0 Request only one remote port specified in remote port lower value.

**Reserved.** A reserved field. It must be x'00'.

## Format of Returned Connection Data

To retrieve the list of TCP/IPv4 connections, request format "NCNN0100 Format," and you will get a repeating list of NCNN0100 tables, each one returning information about a single IPv4 connection. To retrieve the list of TCP/IPv6 connections, request format "NCNN0200 Format" on page 124, and you will get a repeating list of NCNN0200 tables, each one returning information about a single IPv6 connection.

### NCNN0100 Format

The following information about a user space is returned for the NCNN0100 format. For detailed descriptions of the fields in the table, see "Field Descriptions."

Offset		Type	Field
Dec	Hex		
0	0	CHAR(15)	Remote address
15	F	CHAR(1)	Reserved
16	10	BINARY(4)	Remote address binary
20	14	CHAR(15)	Local address
35	23	CHAR(1)	Reserved
36	24	BINARY(4)	Local address binary
40	28	BINARY(4)	Remote port
44	2C	BINARY(4)	Local port
48	30	BINARY(4)	TCP state
52	34	BINARY(4)	Idle time in milliseconds
56	38	BINARY(8)	Bytes in
64	40	BINARY(8)	Bytes out
72	48	BINARY(4)	Connection open type
76	4C	CHAR(10)	Net connection type
86	56	CHAR(2)	Reserved
88	58	CHAR(10)	Associated user profile
98	62	CHAR(2)	Reserved

## Field Descriptions

**Associated user profile.** The user profile of the job on the local system which first performed a sockets API bind() of the socket.

**Note:** This field does not reliably indicate the current user of a connection or socket. To see a list of the jobs or tasks currently using a connection or socket, use the Retrieve Network Connection Data (QtocRtvNetCnnDta) API.

**Bytes in.** The number of bytes received from the remote host.

**Bytes out.** The number of bytes sent to the remote host.

**Connection open type.** The type of open that was done to start this connection. This field only applies to TCP connections.

Possible values are:

- 0                      Passive. A remote host opens the connection.
- 1                      Active. The local system opens the connection.
- 2                      Not supported. Connection open type not supported by protocol.

**Idle time in milliseconds.** The length of time since the last activity on this connection. The length of time is shown in milliseconds.

**Local address.** The local system internet address, in dotted decimal format, of the connection.

**Local address binary.** Binary representation of the local address.

**Local port.** The local system port number.

**Net connection type.** The type of connection or socket. Possible values are:

- \*TCP                    A transmission control protocol (TCP) connection or socket.
  - \*UDP                    A User Datagram Protocol (UDP) socket.
  - \*IPX                    An Internet Protocol (IP) over Internetwork Packet Exchange (IPX) connection or socket.
- Note:** As of V5R2, IP over IPX is no longer supported.
- \*IPS                    An Internet Protocol (IP) over SNA connection or socket.

**Remote address.** The internet address, in dotted decimal format, of the remote host.

The following special value may be returned:

- 0                      This connection is a listening or UDP socket so this field does not apply. The "0" is returned as a left adjusted "0" (x'F0404040...').

**Remote address binary.** Binary representation of the remote address.

The following special value may be returned:

- 0                      This connection is a listening or UDP socket so this field does not apply.

**Remote port.** The remote host port number. Zero is shown if the list entry is for a UDP socket.

**Reserved.** An ignored field.

**TCP state.** A typical connection goes through the states:

- 0                      Listen. Waiting for a connection request from any remote host.

- 1 SYN-sent. Waiting for a matching connection request after having sent connection request.
- 2 SYN-received. Waiting for a confirming connection request acknowledgement.
- 3 Established. The normal state in which data is transferred.
- 4 FIN-wait-1. Waiting for the remote host to acknowledge the local system request to end the connection.
- 5 FIN-wait-2. Waiting for the remote host request to end the connection.
- 6 Close-wait. Waiting for an end connection request from the local user.
- 7 Closing. Waiting for an end connection request acknowledgement from the remote host.
- 8 Last-ACK. Waiting for the remote host to acknowledge an end connection request.
- 9 Time-wait. Waiting to allow the remote host enough time to receive the local system's acknowledgement to end the connection.
- 10 Closed. The connection has ended.
- 11 State value not supported by protocol.

## NCNN0200 Format

The following information about a TCP/IPv6 connection is returned for the NCNN0200 format. For detailed descriptions of the fields in the table, see "Field Descriptions."

Offset		Type	Field
Dec	Hex		
0	0	CHAR(45)	Remote IPv6 address
45	2D	CHAR(3)	Reserved
48	30	CHAR(16)	Remote IPv6 address binary
64	40	CHAR(45)	Local IPv6 address
109	6D	CHAR(3)	Reserved
112	70	CHAR(16)	Local IPv6 address binary
128	80	BINARY(4)	Remote port
132	84	BINARY(4)	Local port
136	88	BINARY(4)	TCP state
140	8C	BINARY(4)	Idle time in milliseconds
144	90	BINARY(8)	Bytes in
152	98	BINARY(8)	Bytes out
160	A0	BINARY(4)	Connection open type
164	A4	CHAR(10)	Net connection type
174	AE	CHAR(10)	Associated user profile

## Field Descriptions

**Associated user profile.** The user profile of the job on the local system which first performed a sockets API bind() of the socket.

**Note:** This field does not reliably indicate the current user of a connection or socket. To see a list of the jobs or tasks currently using a connection or socket, use the Retrieve Network Connection Data (QtocRtvNetCnnDta) API.

**Bytes in.** The number of bytes received from the remote host.

**Bytes out.** The number of bytes sent to the remote host.

**Connection open type.** The type of open that was done to start this connection. This field only applies to TCP connections.

Possible values are:

- 0 Passive. A remote host opens the connection.
- 1 Active. The local system opens the connection.
- 2 Not supported. Connection open type not supported by protocol.

**Idle time in milliseconds.** The length of time since the last activity on this connection. The length of time is shown in milliseconds.

**Local IPv6 address.** The local system internet address, in IPv6 address format, of the connection. This field is NULL padded.

**Local IPv6 address binary.** Binary representation of the local IPv6 address. Even though this field is defined as a character field, a binary IPv6 address is returned in it.

**Local port.** The port number of the local end of the connection.

**Net connection type.** The type of connection or socket.

Possible values are:

- \*TCP A transmission control protocol (TCP) connection or socket.
- \*UDP A User Datagram Protocol (UDP) socket.

**Reserved.** An ignored field.

**Remote IPv6 address.** The internet address, in IPv6 address format, of the remote host. This field is NULL padded.

Special values are:

- :: This connection is a listening socket so this field does not apply.

**Remote IPv6 address binary.** Binary representation of the remote address. Even though this field is defined as a character field, a binary IPv6 address is returned in it.

A special value that may be returned is:

- 0 This connection is a listening socket so this field does not apply. This value is returned as a binary 0.

**Remote port.** The port number of the remote end of the connection.

Special values are:

- 0 This connection is a listening socket so this field does not apply.

**TCP state.** A typical connection goes through the states:

- 0 Listen. Waiting for a connection request from any remote host.
- 1 SYN-sent. Waiting for a matching connection request after having sent connection request.
- 2 SYN-received. Waiting for a confirming connection request acknowledgement.
- 3 Established. The normal state in which data is transferred.
- 4 FIN-wait-1. Waiting for the remote host to acknowledge the local system request to end the connection.
- 5 FIN-wait-2. Waiting for the remote host request to end the connection.

6	Close-wait. Waiting for an end connection request from the local user.
7	Closing. Waiting for an end connection request acknowledgement from the remote host.
8	Last-ACK. Waiting for the remote host to acknowledge an end connection request.
9	Time-wait. Waiting to allow the remote host enough time to receive the local system's acknowledgement to end the connection.
10	Closed. The connection has ended.
11	State value not supported by protocol.

## Error Messages

Message ID	Error Message Text
TCP84C0 E	TCP/IP stack not active.
TCP84C5 E	Error providing TCP/IP Network Status list information.
TCP84C6 E	Internal operations error - RESULT &1 CC &2 RC &3 ERRNO &4.
TCP84C7 E	Connections list qualifier parameter not valid.
CPF0F03 E	Error in retrieving the user space that was created by the caller.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1E E	Required parameter &1 omitted.
CPF3C21 E	Format name &1 is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	API contains a problem. See prior messages to determine why the failure occurred.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R1

[Top](#) | ["Communications APIs," on page 1](#) | [APIs by category](#)

---

## List Network Interfaces (QtocLstNetlfc) API

Required Parameter Group:	
<b>1</b>	Qualified user space name
<b>Input</b>	Char(20)
<b>2</b>	Format name
<b>Input</b>	Char(8)
<b>3</b>	Error Code
<b>I/O</b>	Char(*)
Service Program: QTOCNETSTS	
Threadsafe: Yes	

The List Network Interfaces (QtocLstNetIfc) API returns a list of all logical TCP/IP interfaces with details. This API returns all IPv4 logical interfaces using one output format name, and all IPv6 logical interfaces using a different output format name.

TCP/IP must be active; otherwise error message TCP84C0 will be issued.

## Authorities and Locks

*User Space Library Authority*  
\*EXECUTE

*User Space Authority*  
\*CHANGE

*User Space Lock*  
\*SHRNUP

## Required Parameter Group

**Qualified user space name**  
INPUT; CHAR(20)

The user space that is to receive the created list. The first 10 characters contain the user space name, and the second 10 characters contain the name of the library in which the user space is located. You can use these special values for the library name:

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

**Format name**  
INPUT; CHAR(8)

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

NIFC0100            Detailed information about each TCP/IPv4 network interface. Refer to "NIFC0100 Format" on page 128 for details on the format.  
NIFC0200            Detailed information about each TCP/IPv6 network interface. Refer to "NIFC0200 Format" on page 133 for details on the format.

**Error code**  
I/O; CHAR(\*)

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

## Format of Interface Lists

To request a list of all logical interfaces, use format NIFC0100. The interface description list consists of:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:
  - NIFC0100 format, or
  - NIFC0200 format

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections.

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API Examples.

## Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name specified

## Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name used
10	A	CHAR(10)	User space library name used

## Format of Returned Connection Data

To retrieve the list of TCP/IPv4 network interfaces, request format “NIFC0100 Format,” and you will get a repeating list of NIFC0100 tables, each one returning information about a single IPv4 network interface. To retrieve the list of TCP/IPv6 network interfaces, request format “NIFC0200 Format” on page 133, and you will get a repeating list of NIFC0200 tables, each one returning information about a single IPv6 network interface.

## NIFC0100 Format

The following information about each TCP/IPv4 logical interface is returned for the NIFC0100 format. For detailed descriptions of the fields in the table, see “Field Descriptions” on page 129.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(15)	Internet address
15	F	CHAR(1)	Reserved
16	10	BINARY(4)	Internet address binary
20	14	CHAR(15)	Network address
35	23	CHAR(1)	Reserved
36	24	BINARY(4)	Network address binary
40	28	CHAR(10)	Network name
50	32	CHAR(10)	Line description
60	3C	CHAR(10)	Interface name
70	46	CHAR(2)	Reserved
72	48	BINARY(4)	Interface status
76	4C	BINARY(4)	Interface type of service
80	50	BINARY(4)	Interface MTU



Offset		Type	Field
Dec	Hex		
84	54	BINARY(4)	Interface line type
88	58	CHAR(15)	Host address
103	67	CHAR(1)	Reserved
104	68	BINARY(4)	Host address binary
108	6C	CHAR(15)	Interface subnet mask
123	7B	CHAR(1)	Reserved
124	7C	BINARY(4)	Interface subnet mask binary
128	80	CHAR(15)	Directed broadcast address
143	8F	CHAR(1)	Reserved
144	90	BINARY(4)	Directed broadcast address binary
148	94	CHAR(8)	Change date
156	9C	CHAR(6)	Change time
162	A2	CHAR(15)	Associated local interface
177	B1	CHAR(3)	Reserved
180	B4	BINARY(4)	Associated local interface binary
184	B8	BINARY(4)	Change status
188	BC	BINARY(4)	Packet rules
192	C0	BINARY(4)	Automatic start
196	C4	BINARY(4)	TRLAN bit sequencing
200	C8	BINARY(4)	Interface type
204	CC	BINARY(4)	Proxy ARP enabled
208	D0	BINARY(4)	Proxy ARP allowed
212	D4	BINARY(4)	Configured MTU
216	D8	CHAR(24)	Network name - full
240	F0	CHAR(24)	Interface name - full

## Field Descriptions

**Associated local interface.** The internet address, in dotted decimal notation, of the local interface that has been associated with this interface. The following is a special value:

\*NONE No association has been made between this interface and another local interface.

**Associated local interface binary.** Binary representation of the associated local interface. The following is a special value:

0 No association has been made between this interface and another local interface.

**Automatic start.** Whether the interface is started automatically when the TCP/IP stack is activated. Possible values are:

0 NO. This interface is not started automatically.

1 YES. This interface is started automatically.

**Change date.** The date of the most recent change to this interface in the dynamic tables used by the TCP/IP protocol stack. It is returned as 8 characters in the form YYYYMMDD, where:

YYYY	Year
MM	Month
DD	Day

**Change status.** The status of the most recent change to this interface in the dynamic tables used by the TCP/IP protocol stack.

1	Add interface request processed
2	Change interface request processed
3	Start interface request processed
4	End interface request processed

**Change time.** The time of the most recent change to this interface in the dynamic tables used by the TCP/IP protocol stack. It is returned as 6 characters in the form HHMMSS, where:

HH	Hour
MM	Minute
SS	Second

**Configured MTU.** The configured maximum transmission unit value specified for this interface. The following is a special value:

0 LIND. The interface is not active currently and the MTU was specified as \*LIND.

**Directed broadcast address.** The internet address, in dotted decimal notation, used to broadcast to all systems attached to the same network or subnetwork as this interface. The following is a special value:

\*NONE The interface is attached to a network that does not support a broadcast operation.

**Directed broadcast address binary.** Binary representation of the directed broadcast address. The following is a special value:

0 The interface is attached to a network that does not support a broadcast operation.

**Host address.** Host portion of the internet address, in dotted decimal notation, as determined by the subnet mask specified for this interface.

**Host address binary.** Binary representation of the host address.

**Interface line type.** Type of line used by an interface. The following link protocols are supported:

-1	OTHER - IPI - An Internet Protocol (IP) over Internetwork Pack Exchange (IPX) interface. IPS - An Internet Protocol (IP) over SNA interface.  <b>Note:</b> As of V5R2, IP over IPX is no longer supported.
-2	NONE - Line is not defined. This is used for the following interfaces: *LOOPBACK, *VIRTUALIP, *OPC. There is no line type value for these interfaces.
-3	ERROR - This value is displayed if any system errors other than those for *NOTFND are received while trying to determine the link type for an interface.
-4	NOTFND - Not found. This value is displayed if the line description object for this interface cannot be found.
1	ELAN - Ethernet local area network protocol.
2	TRLAN - Token-ring local area network protocol.
3	FR - Frame relay network protocol.
4	ASYNC - Asynchronous communications protocol.

- 5 PPP - Point-to-point Protocol.
- 6 WLS - Wireless local area network protocol.
- 7 X.25 - X.25 protocol.
- 8 DDI - Distributed Data Interface protocol.
- 9 TDLC - Twinaxial Datalink Control. Used for TCP/IP over Twinax.
- 10 L2TP (Virtual PPP) - Layer Two Tunneling Protocol.

**Interface MTU.** Maximum transmission unit value specified for this interface. The following are special values:

- 1 OTHER.  
 IPI - An Internet Protocol (IP) over Internetwork Pack Exchange (IPX) interface.  
 IPS - An Internet Protocol (IP) over SNA interface.  
  
**Note:** As of V5R2, IP over IPX is no longer supported.
- 0 LIND - The interface is not active currently and the MTU was specified as \*LIND.

**Interface name.** The first 10 characters of the name of this interface.

**Interface name - full.** The complete 24 character interface name.

**Interface status.** Current status of this logical interface.

- 0 Inactive - The interface has not been started. The interface is not active.
- 1 Active - The interface has been started and is running.
- 2 Starting - The system is processing the request to start this interface.
- 3 Ending - The system is processing the request to end this interface.
- 4 RCYPND - An error with the physical line associated with this interface was detected by the system. The line description associated with this interface is in the recovery pending (RCYPND) state.
- 5 RCYCNL - A hardware failure has occurred and the line description associated with this interface is in the recovery canceled (RCYCNL) state.
- 6 Failed - The line description associated with this interface has entered the failed state.
- 7 Failed (TCP) - An error was detected in the IBM TCP/IP Vertical Licensed Internal Code.
- 8 DOD - Point-to-Point (PPP) Dial-on-Demand.

**Interface subnet mask.** The subnet mask for the network, subnet, and host address fields of the internet address, in dotted decimal notation, that defines the subnetwork for an interface.

**Interface subnet mask binary.** Binary representation of the interface subnet mask.

**Interface type.** The interface types are:

- 0 Broadcast capable
- 1 Non-broadcast capable
- 2 Unnumbered network

**Interface type of service.** The way in which the internet hosts and routers should make trade-offs between throughput, delay, reliability and cost. The following are special values:

- 1 OTHER -  
 IPI - An Internet Protocol (IP) over Internetwork Pack Exchange (IPX) interface.  
 IPS - An Internet Protocol (IP) over SNA interface.  
  
**Note:** As of V5R2, IP over IPX is no longer supported.
- 1 NORMAL - Used for delivery of datagrams.
- 2 MINDELAY - Prompt delivery of datagrams with the minimize delay indication.
- 3 MAXTHRPUT - Datagrams with maximize throughput indication.

- 4 MAXRLB - Datagrams with maximize reliability indication.
- 5 MINCOST - Datagram with minimize monetary cost indication.

**Internet address.** The internet address, in dotted decimal notation, of an interface.

**Internet address binary.** Binary representation of the internet address.

**Line description** Name of the communications line description that identifies the physical network associated with an interface. The following are special values:

- \**IPI* This interface is used by Internet Protocol (IP) over Internetwork Packet Exchange (IPX).  
**Note:** As of V5R2, IP over IPX is no longer supported.
- \**IPS* This interface is used by Internet Protocol (IP) over SNA.
- \**LOOPBACK* This is a loopback interface. Processing associated with a loopback interface does not extend to a physical line.
- \**VIRTUALIP* The virtual interface is a circuitless interface. It is used in conjunction with the associated local interface (LCLIFC) when adding standard interfaces.
- \**OPC* This interface is attached to the optical bus (OptiConnect).

**Network address.** Internet address, in dotted decimal notation, of the IP network or subnetwork to which the interface is attached.

**Network address binary.** Binary representation of the network address.

**Network name.** The first 10 characters of the name of the network that this interface is a part of.

**Network name - full.** The complete 24 character name of the network that this interface is a part of.

**Packet rules.** The kind of packet rules data available for a particular line.

- 1 OTHER - An unknown Packet rules value.
- 0 None - No filters and no NAT are loaded for the line specified.
- 1 NAT - NAT is enabled for this line.
- 2 Filters - Filters are defined for this line.
- 3 Filters and NAT - NAT enabled and filters defined.
- 4 Filters and IPsec - Filters and IPsec filters are defined for this line.
- 5 NAT and Filters and IPsec - NAT enabled and Filters and IPsec filters defined.

**Proxy ARP enabled.** Whether Proxy ARP is currently active for this interface. Proxy ARP allows physically distinct separate networks to appear as if they are a single logical network. It provides connectivity between physically separate network without creating any new logical networks and without updating any route tables.

- 0 NO - Proxy ARP not enabled.
- 1 YES - Proxy ARP enabled.

**Proxy ARP allowed.** This field applies to Opticonnect (\*OPC) and Virtual Ethernet interfaces only. For those types of interfaces, this field indicates whether Proxy ARP has been configured to be allowed or not allowed.

- 0 NO - Proxy ARP not allowed.
- 1 YES - Proxy ARP allowed.
- 2 Unsupported - Proxy ARP allowed field is not supported by this interface.

**Reserved.** An ignored field.

**TRLAN bit sequencing.** The order the Address Resolution Protocol (ARP) puts bits into the hardware address for Token Ring. Possible values are:

- 1 MSB - The most significant bit is placed first.
- 2 LSB - The least significant bit is placed first.

## NIFC0200 Format

The following information about each TCP/IPv6 logical interface is returned for the NIFC0200 format. For detailed descriptions of the fields in the table, see “Field Descriptions” on page 134.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(45)	Internet IPv6 address
45	2D	CHAR(3)	Reserved
48	30	CHAR(16)	Internet IPv6 address binary
64	40	CHAR(3)	Interface prefix length
67	43	CHAR(1)	Reserved
68	44	BINARY(4)	Interface prefix length binary
72	48	BINARY(4)	Address type
76	4C	BINARY(4)	Address state
80	50	BINARY(8)	Address preferred lifetime
88	58	CHAR(8)	Address preferred lifetime expiration date
96	60	CHAR(6)	Address preferred lifetime expiration time
102	66	CHAR(2)	Reserved
104	68	BINARY(8)	Address valid lifetime
112	70	CHAR(8)	Address valid lifetime expiration date
120	78	CHAR(6)	Address valid lifetime expiration time
126	7E	CHAR(10)	Line name
136	88	BINARY(4)	Interface line type
140	8C	CHAR(50)	Interface description
190	BE	CHAR(45)	Network IPv6 address
235	EB	CHAR(1)	Reserved
236	EC	CHAR(16)	Network IPv6 address binary
252	FC	CHAR(45)	Host IPv6 address
297	129	CHAR(3)	Reserved
300	12C	CHAR(16)	Host IPv6 address binary
316	13C	BINARY(4)	Interface status
320	140	BINARY(4)	Automatic start
324	144	BINARY(4)	Packet rules
328	148	BINARY(4)	Interface source
332	14C	BINARY(4)	Duplicate address detection transmits
336	150	BINARY(4)	Multicast - number of references
340	154	CHAR(4)	Reserved
344	158	CHAR(8)	Change date
352	160	CHAR(6)	Change time
358	166	CHAR(2)	Reserved

## Field Descriptions

**Address preferred lifetime.** The length of time that a "valid" address is preferred, in seconds. When the preferred lifetime expires, the address becomes Deprecated. See the Address State field description for more information. Valid values range from -31536000 through 4294967295 seconds. Negative values indicate that the Address preferred lifetime expired that number of seconds ago.

The following are special values:

-1000000000      Infinite - this address has an infinite preferred lifetime.  
-1000000001      Not Applicable - this address is not in the Preferred address state, so this field does not apply.

**Address preferred lifetime expiration date.** The date when this address will no longer be preferred. If the Address preferred lifetime expiration date and time are in the future, the address is still preferred. If the Address preferred lifetime expiration date and time are in the past, then this address is no longer preferred. The Address preferred lifetime expiration date is returned as 8 characters in the form YYYYMMDD.

The meaning of the characters is as follows:

YYYY      Year  
MM      Month  
DD      Day

The following are special values:

00000000      Infinite - this address has an infinite preferred lifetime which never expires.  
00000001      Not Applicable - this address is not in the Preferred address state, so this field does not apply.

**Address preferred lifetime expiration time.** The time when this address will no longer be in the preferred state. If the Address preferred lifetime expiration date and time are in the future, the address is still preferred. If the Address preferred lifetime expiration date and time are in the past, then this address is no longer preferred. The Address preferred lifetime expiration time is returned as 6 characters in the form HHMMSS.

The meaning of the characters is as follows:

HH      Hour  
MM      Minute  
SS      Second

The following are special values:

000000      Infinite - this address has an infinite preferred lifetime which never expires.  
000001      Not Applicable - this address is not in the preferred address state, so this field does not apply.

**Address state.** The current state of this IPv6 address. IPv6 addresses are in different states at different times, due to Duplicate Address Detection (DAD) and address lifetimes. Unicast and Multicast addresses have different possible states and the only state that applies to either type is Failed.

When a unicast address is one of the two "valid" states, Preferred and Deprecated, it may be used as the source or destination address of a packet. When a unicast address is in one of the five "invalid" states it may not be used as the source or destination address of a packet. The five "invalid" states are: Tentative, Expired, Inactive, Duplicate and Failed.

When a multicast address is one of the two "valid" states, Idle Listener and Delaying Listener, it may be used as the source or destination address of a packet. When a multicast address is in one of the two "invalid" states, Non-listener and Failed, it may not be used as the source or destination address of a packet.

Possible values are:

- 1 Failed - while attempting to move this address from one state to another, an internal error occurred, preventing completion of the action necessary to perform the state change.
- 1 Inactive - the interface has been ended by the user and no further communications will be performed using this address. The address is available to be reassigned elsewhere.
- 2 Duplicate - A duplicate address was detected on the network during Duplicate Address Detection (DAD), therefore this address was not moved to the Preferred state.
- 3 Tentative - an address whose uniqueness on a link is being verified, prior to its assignment to a physical interface. A tentative address is not considered assigned to a physical interface in the usual sense. A physical interface discards received packets addressed to a tentative address, but accepts Neighbor Discovery packets related to Duplicate Address Detection for the tentative address.
- 4 Preferred - an address assigned to a physical interface whose use by upper layer protocols is unrestricted. Preferred addresses may be used as the source (or destination) address of packets sent from (or to) the physical interface.
- 5 Deprecated - an address assigned to a physical interface whose use is discouraged, but not forbidden. A deprecated address should no longer be used as a source address in new communications, but packets sent from or to deprecated addresses are delivered as expected. A deprecated address may continue to be used as a source address in communications where switching to a preferred address causes hardship to a specific upper-layer activity (for example, an existing TCP connection).
- 6 Expired - an address assigned to a physical interface whose use is forbidden. An address transitions to the expired state when its valid lifetime expires. An IPv6 interface with an expired address will be removed after a period of time.
- 11 Non-listener - the initial state of a multicast address when it first joins a multicast group and it is not yet listening for any incoming Multicast Listener Discovery requests.
- 12 Idle listener - this multicast interface is listening for incoming Multicast Listener Discovery requests.
- 13 Delaying listener - this multicast interface has received an incoming Multicast Listener Discovery request and has gone to sleep until it is time to wakeup and take action on that request.

**Address type.** The type of IPv6 address that is assigned to this network interface.

Possible values are:

- 1 Unicast - an identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.
- 2 Multicast - an identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.
- 3 Anycast - an identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance).

**Address valid lifetime.** The length of time, in seconds, that an address remains in a "valid" state (Preferred or Deprecated). When the valid lifetime expires, the address becomes Expired. See the Address State field description for more information. Valid values range from -31536000 through 4294967295 seconds. Negative values indicate that the Address valid lifetime expired that number of seconds ago.

The following are special values:

- 1000000000 Infinite - this address has an infinite valid lifetime.
- 1000000001 Not Applicable - this address is not in a valid address state, so this field does not apply.

**Address valid lifetime expiration date.** The date when this address will expire or did expire. If the Address valid lifetime expiration date and time are in the future, the address has not expired yet. If the Address valid lifetime expiration date and time are in the past, then this address has expired and is still

being returned for a short period of time to indicate that the interface ceased to function because its valid lifetime expired. The Address valid lifetime expiration date is returned as 8 characters in the form YYYYMMDD.

The meaning of the characters is as follows:

YYYY	Year
MM	Month
DD	Day

The following are special values:

00000000	Infinite - this address has an infinite valid lifetime which never expires.
00000001	Not Applicable - this address is not in a valid address state, so this field does not apply.

**Address valid lifetime expiration time.** The time when this address will expire or did expire. If the Address valid lifetime expiration date and time are in the future, the address has not expired yet. If the Address valid lifetime expiration date and time are in the past, then this address has expired and is still being returned for a short period of time to indicate that the interface ceased to function because its valid lifetime expired. The Address valid lifetime expiration time is returned as 6 characters in the form HHMMSS.

The meaning of the characters is as follows:

HH	Hour
MM	Minute
SS	Second

The following are special values:

000000	Infinite - this address has an infinite valid lifetime which never expires.
000001	Not Applicable - this address is not in a valid address state, so this field does not apply.

**Automatic start.** Whether the interface is started automatically when the TCP/IPv6 stack is activated. Possible values are:

0	NO. This interface is not started automatically.
1	YES. This interface is started automatically.

**Change date.** The date of the most recent change to this interface in the dynamic tables used by the TCP/IPv6 protocol stack. It is returned as 8 characters in the form YYYYMMDD, where:

YYYY	Year
MM	Month
DD	Day

**Change time.** The time of the most recent change to this interface in the dynamic tables used by the TCP/IPv6 protocol stack. It is returned as 6 characters in the form HHMMSS, where:

HH	Hour
MM	Minute
SS	Second



**Duplicate address detection transmits.** Specifies the number of Duplicate Address Detection (DAD) transmissions the stack has sent out on this interface.

**Host IPv6 address.** Host portion of the internet address, in IPv6 address format, as determined by the prefix length configured for this interface. This field is NULL padded.

**Host IPv6 address binary.** Binary representation of the host IPv6 address. Even though this field is defined as a character field, a binary IPv6 address is returned in it.

**Interface description** Configured free form comment field about this interface.

**Interface line type.** Type of line used by the interface. The following link protocols are supported:

-1	OTHER
-2	NONE - Line is not defined. This value is used for the following interfaces: *LOOPBACK6, *VIRTUALIP, *OPC. There is no line type value for these interfaces.
-3	ERROR - This value is displayed if any system errors other than those for *NOTFND are received while trying to determine the link type for an interface.
-4	NOTFND - Not found. This value is displayed if the line description object for this interface cannot be found.
1	ELAN - Ethernet local area network protocol.
2	TRLAN - Token-ring local area network protocol.
3	FR - Frame relay network protocol.
4	ASYNC - Asynchronous communications protocol.
5	PPP - Point-to-point Protocol.
6	WLS - Wireless local area network protocol.
7	X.25 - X.25 protocol.
8	DDI - Distributed Data Interface protocol.
9	TDLC - Twinaxial Datalink Control. Used for TCP/IP over Twinax.
10	L2TP (Virtual PPP) - Layer Two Tunneling Protocol.
11	IPv6 Tunneling Line - Any kind of IPv6 over IPv4 tunnel.

**Interface prefix length.** The prefix length defines how many bits of the interface IPv6 address are in the prefix. It is a zoned decimal number which specifies how many of the left-most bits of the address make up the prefix. The prefix length is used to generate network and host addresses. This field is NULL padded.

**Interface prefix length binary.** Binary representation of the interface prefix length.

**Interface source** Specifies how this interface was added to the TCP/IPv6 stack.

Possible values are:

1	Stateless - the interface was added to the stack by the IPv6 stateless autoconfiguration mechanism.
2	Stateful - the interface was added to the stack by the IPv6 stateful configuration mechanism (that is, DHCPv6).
3	Manual - the interface was added to the stack by manual configuration.

**Interface status.** Current status of this logical interface.

0	Inactive - The interface has not been started. The interface is not active.
1	Active - The interface has been started and is running.
2	Starting - The system is processing the request to start this interface.
3	Ending - The system is processing the request to end this interface.
4	RCYPND - An error with the physical line associated with this interface was detected by the system. The line description associated with this interface is in the recovery pending (RCYPND) state.
5	RCYCNL - A hardware failure has occurred and the line description associated with this interface is in the recovery canceled (RCYCNL) state.
6	Failed - The line description associated with this interface has entered the failed state.

**Internet IPv6 address.** The internet address, in IPv6 address format, of the interface.

**Internet IPv6 address binary.** Binary representation of the internet IPv6 address. Even though this field is defined as a character field, a binary IPv6 address is returned in it.

**Line name.** Name of the communications line description that identifies the physical network associated with an interface. This field is NULL padded.

The following are special values:

<i>*LOOPBACK6</i>	This is the IPv6 loopback interface. Processing associated with a loopback interface does not extend to a physical line.
<i>*VIRTUALIP</i>	The virtual interface is a circuitless interface. It is used in conjunction with the associated local interface (LCLIFC) when adding standard interfaces.
<i>*OPC</i>	This interface is attached to the optical bus (OptiConnect).
<i>*TNLCFG64</i>	This interface is associated with a configured 6-4 tunneling line.

**Multicast - number of references.** The number of Sockets clients that have joined this multicast group.

The following is a special value:

-1	This interface is not a Multicast address and this field does not apply.
----	--

**Network IPv6 address.** Internet address, in IPv6 address format, of the IPv6 network or subnetwork to which the interface is attached. This field is NULL padded.

**Network IPv6 address binary.** Binary representation of the network IPv6 address. Even though this field is defined as a character field, a binary IPv6 address is returned in it.

**Packet rules.** The kind of packet rules data available for the particular line this interface is associated with.

-1	OTHER - An unknown Packet rules value.
0	None - No filters and no NAT are loaded for the line specified.
1	NAT - NAT is enabled for this line.
2	Filters - Filters are defined for this line.
3	Filters and NAT - NAT enabled and filters defined.
4	Filters and IPsec - Filters and IPsec filters are defined for this line.
5	NAT and Filters and IPsec - NAT enabled and Filters and IPsec filters defined.

**Reserved.** An ignored field.

## Error Messages

Message ID	Error Message Text
TCP84C0 E	TCP/IP stack not active.
TCP84C5 E	API error listing TCP/IP Network Status list information.
TCP84C6 E	Internal operations error - RESULT &1 CC &2 RC &3 ERRNO &4.
TCP84C9 I	Information returned incomplete.
CPF0F03 E	Error in retrieving the user space that was created by the caller.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1E E	Required parameter &1 omitted.
CPF3C21 E	Format name &1 is not valid.

Message ID	Error Message Text
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	API contains a problem. See prior messages to determine why the failure occurred.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R1

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

## List Network Routes (QtocLstNetRte) API

Required Parameter Group:	
1	Qualified user space name
<b>Input</b>	Char(20)
2	Format name
<b>Input</b>	Char(8)
3	Error Code
<b>I/O</b>	Char(*)
	Service Program: QTOCNETSTS
Threadsafe: Yes	

The List Network Routes (QtocLstNetRte) API returns a detailed list of all routes. This API returns all IPv4 routes using one output format name, and all IPv6 routes using a different output format name.

TCP/IP must be active on this system; otherwise, a message will be issued.

### Authorities and Locks

*User Space Library Authority*  
\*EXECUTE

*User Space Authority*  
\*CHANGE

*User Space Lock*  
\*SHRNUP

### Required Parameter Group

**Qualified user space name**  
INPUT; CHAR(20)

The user space that is to receive the created list. The first 10 characters contain the user space name, and the second 10 characters contain the name of the library where the user space is located. You can use these special values for the library name:

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

### Format name

INPUT; CHAR(8)

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

*NRTE0100*            Detailed information about each TCP/IPv4 route. Refer to "NRTE0100 Format" on page 141 for details on the format.  
*NRTE0200*            Detailed information about each TCP/IPv6 route. Refer to "NRTE0200 Format" on page 145 for details on the format.

### Error code

I/O; CHAR(\*)

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

## Format of Route Lists

To request a list of all routes, use format NRTE0100.

The route description list consists of:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:
- - NRTE0100 format, or
  - NRTE0200 format<

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections.

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API Examples.

## Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name specified

## Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name used
10	A	CHAR(10)	User space library name used

## Format of Returned Connection Data

To retrieve the list of TCP/IPv4 routes, request format “NRTE0100 Format,” and you will get a repeating list of NRTE0100 tables, each one returning information about a single IPv4 route. To retrieve the list of TCP/IPv6 routes, request format “NRTE0200 Format” on page 145, and you will get a repeating list of NRTE0200 tables, each one returning information about a single IPv6 route.

## NRTE0100 Format

The following information about each TCP/IPv4 route is returned for the NRTE0100 format. For detailed descriptions of the fields in the table, see “Field Descriptions” on page 142.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(15)	Route destination
15	F	CHAR(1)	Reserved
16	10	BINARY(4)	Route destination binary
20	14	CHAR(15)	Subnet mask
35	23	CHAR(1)	Reserved
36	24	BINARY(4)	Subnet mask binary
40	28	CHAR(15)	Next hop
55	37	CHAR(1)	Reserved
56	38	BINARY(4)	Next hop binary
60	3C	BINARY(4)	Route status
64	40	BINARY(4)	Type of service
68	44	BINARY(4)	Route MTU
72	48	BINARY(4)	Route type
76	4C	BINARY(4)	Route source
80	50	BINARY(4)	Route precedence
84	54	BINARY(4)	Local binding interface status
88	58	BINARY(4)	Local binding type
92	5C	BINARY(4)	Local binding line type
96	60	CHAR(15)	Local binding interface
111	6F	CHAR(1)	Reserved
112	70	BINARY(4)	Local binding interface binary
116	74	CHAR(15)	Local binding subnet mask
131	83	CHAR(1)	Reserved
132	84	BINARY(4)	Local binding subnet mask binary
136	88	CHAR(15)	Local binding network address
151	97	CHAR(1)	Reserved
152	98	BINARY(4)	Local binding network address binary
156	9C	CHAR(10)	Local binding line description

Offset		Type	Field
Dec	Hex		
166	A6	CHAR(8)	Change date
174	AE	CHAR(6)	Change time

## Field Descriptions

**Change date.**The date of the most recent change to this route in the dynamic tables used by the TCP/IP protocol stack. It is returned as 8 characters in the form YYYYMMDD, where:

YYYY            Year  
MM              Month  
DD              Day

**Change time.**The time of the most recent change to this route in the dynamic tables used by the TCP/IP protocol stack. It is returned as 6 characters in the form HHMMSS, where:

HH              Hour  
MM              Minute  
SS              Second

**Local binding interface.** The IP interface to bind to this route.

**Local binding interface binary.** Binary representation of the local binding interface.

**Local binding interface status.** The current status of this logical interface.

The possible values are:

0                Inactive - The interface has not been started. The interface is not active.  
1                Active - The interface has been started and is running.  
2                Starting - The system is processing the request to start this interface.  
3                Ending - The system is processing the request to end this interface.  
4                RCYPND - An error with the physical line associated with this interface was detected by the system. The line description associated with this interface is in the recovery pending (RCYPND) state.  
5                RYCYNL - A hardware failure has occurred and the line description associated with this interface is in the recovery canceled (RYCYNL) state.  
6                Failed - The line description associated with this interface has entered the failed state.  
7                Failed (TCP) - An error was detected in the IBM TCP/IP Vertical Licensed Internal Code.  
8                DOD - Point-to-Point (PPP) Dial-on-Demand.  
9                Active Duplicate IP Address Conflict - Another host on the LAN responded to a packet destined for this logical interface.

**Local binding line description.** Each TCP/IP interface is associated with a network. This field displays the name of the communications line description or virtual line (L2TP) that identifies the network associated with an interface. The following are special values:

\**IPI*            This interface is used by Internet Protocol (IP) over Internetwork Packet Exchange (IPX)  
  
                  **Note:** As of V5R2, IP over IPX is no longer supported.  
                  This interface is used by Internet Protocol (IP) over SNA.  
\**LOOPBACK*    This is a loopback interface. Processing associated with a loopback interface does not extend to a physical line.  
\**VIRTUALIP*    The virtual interface is a circuitless interface. It is used in conjunction with the associated local interface (LCLIFC) when adding standard interfaces.

\*OPC

4 This interface is attached to the optical bus (OptiConnect).

**Local binding line type.** Indicates the type of line used by an interface. The following link protocols are supported:

- 1 OTHER -
  - IPI - An Internet Protocol (IP) over Internetwork Pack Exchange (IPX).
  - IPS - An Internet Protocol (IP) over SNA interface.
- 2 NONE - Line is not defined. This is used for the following interfaces: \*LOOPBACK, \*VIRTUALIP, \*OPC. There is no line type value for these interfaces.
- 3 ERROR - This value is displayed if any system errors other than those for \*NOTFND are received while trying to determine the link type for an interface.
- 4 NOTFND - Not found. This value is displayed if the line description object for this interface cannot be found.
- 1 ELAN - Ethernet local area network protocol.
- 2 TRLAN - Token-ring local area network protocol.
- 3 FR - Frame relay network protocol.
- 4 ASYNC - Asynchronous communications protocol.
- 5 PPP - Point-to-point Protocol.
- 6 WLS - Wireless local area network protocol.
- 7 X.25 - X.25 protocol.
- 8 DDI - Distributed Data Interface protocol.
- 9 TDLC - Twinaxial Datalink Control. Used for TCP/IP over Twinax.

**Local binding network address.** The internet address, in dotted decimal notation, of the IP network or subnetwork that the interface is attached to.

**Local binding network address binary.** Binary representation of the local binding network address.

**Local binding subnet mask.** The subnet mask for the network, subnet, and host address fields of the internet address, in dotted decimal notation, that defines the subnetwork for an interface.

**Local binding subnet mask binary.** Binary representation of the local binding subnet mask.

**Local binding type.** The possible values are:

- 0 Dynamic
- 1 Static

**Next hop.** The internet address of the first system on the path from your system to the route destination in dotted decimal notation. The following are special values:

\*DIRECT

This is the next hop value of a route that is automatically created. When an interface is added to this system, a route to the network the interface attaches to is also created.

**Next hop binary.** The binary representation of the next hop. For \*DIRECT this will be the local binding network address.

**Reserved.** An ignored field.

**Route destination.** The Internet Protocol (IP) address, in dotted decimal notation, of the ultimate destination reached by this route. When used in combination with the subnet mask and the type of service values, the route destination identifies a route to a network or system.

**Route destination binary.** The binary representation of the route destination.

**Route MTU.** A number representing the maximum transmission unit (MTU) value for this route in bytes. The following are special values:

- 1 OTHER -
  - IPI - An Internet Protocol (IP) over Internetwork Pack Exchange (IPX) interface.
  - IPS - An Internet Protocol (IP) over SNA interface.
- 0 **Note:** As of V5R2, IP over IPX is no longer supported.  
IFC - The route is not currently active and the MTU was specified as \*IFC.

**Route precedence.** Identify priority of route, range 1-10. Lowest priority being 1.

**Route source.** Specifies how this route was added to the IP routing tables. The possible values are:

- 1 OTHER - The route was added by a sockets input/output control (IOctl) or other mechanism.
- 0 CFG - The route was added with system configuration commands.
- 1 ICMP - The route was added by the Internet Control Message Protocol (ICMP) redirect mechanism.
- 2 SNMP - The route was added by the Simple Network Management Protocol (SNMP).
- 3 RIP - The route was added by the Routing Information Protocol (RIP).

**Route status.** Indicated whether this route is available.

- 1 YES - The router specified by the next hop value for this interface is available for use. This route is included among the routes considered when datagram routing is performed by TCP/IP.
- 2 NO - The router specified by the next hop value for this interface is not available for use, interface is not active. This route is not included among the routes considered when datagram routing is performed.
- 3 DOD - This route is used for Point-to-Point (PPP) Dial-on-Demand. Currently, this Dial-on-Demand route is not available. The route will become available when a Dial-on-Demand session is initiated for the interface this route is associated with.
- 4 NO GATEWAY - The router specified by the next hop value for this interface is not available for use, the router may be experiencing a problem.

**Route type.** The route types are:

- 0 DFTRROUTE - A default route.
- 1 DIRECT - A route to a network or subnetwork to which this system has a direct physical connection.
- 2 HOST - A route to a specific remote host.
- 3 SUBNET - An indirect route to a remote subnetwork.
- 4 NET - An indirect route to a remote network.

**Subnet mask.** The actual value of the subnet mask in dotted decimal notation.

**Subnet mask binary.** The binary representation of the subnet mask.

**Type of service.** Defines how the internet hosts and routers should make trade-offs between throughput, delay, reliability and cost. The following are special values:

- 1 OTHER -
  - IPI - An Internet Protocol (IP) over Internetwork Pack Exchange (IPX) interface.
  - IPS - An Internet Protocol (IP) over SNA interface.
- 1 **Note:** As of V5R2, IP over IPX is no longer supported.  
NORMAL - Used for delivery of datagrams.
- 2 MINDELAY - Prompt delivery of datagrams with the minimize delay indication.
- 3 MAXTHRPUT - Datagrams with maximize throughput indication.



- 4 MAXRLB - Datagrams with maximize reliability indication.
- 5 MINCOST - Datagrams with minimize monetary cost indication.

## NRTE0200 Format

The following information about each TCP/IPv6 route is returned for the NRTE0200 format. For detailed descriptions of the fields in the table, see “Field Descriptions.”

Offset		Type	Field
Dec	Hex		
0	0	CHAR(45)	Route destination
45	2D	CHAR(3)	Reserved
48	30	CHAR(16)	Route destination binary
64	40	CHAR(3)	Prefix length
67	43	CHAR(5)	Reserved
72	48	BINARY(4)	Prefix length binary
76	4C	BINARY(4)	Next hop address family
80	50	CHAR(45)	Next hop IPv6
125	7D	CHAR(3)	Reserved
128	80	CHAR(16)	Next hop IPv6 binary
144	90	CHAR(15)	Next hop IPv4
159	9F	CHAR(1)	Reserved
160	A0	BINARY(4)	Next hop IPv4 binary
164	A4	CHAR(10)	Local binding line name
174	AE	CHAR(2)	Reserved
176	B0	BINARY(4)	Local binding line type
180	B4	BINARY(4)	Local binding line status
184	B8	BINARY(4)	Route status
188	BC	BINARY(8)	Route lifetime remaining
196	C4	BINARY(4)	Route lifetime at creation
200	C8	BINARY(4)	Route source
204	CC	BINARY(4)	Route type
208	D0	BINARY(4)	Configured route MTU
212	D4	BINARY(4)	Actual route MTU
216	D8	BINARY(4)	Is on-link
220	DC	BINARY(4)	Duplicate indicator
224	E0	CHAR(8)	Change date
232	E8	CHAR(6)	Change time
238	EE	CHAR(8)	Expiration date
246	F6	CHAR(6)	Expiration time

## Field Descriptions

**Actual route MTU.** A number representing the maximum transmission unit (MTU) value for this route in bytes.

The following is a special value:

0 \*IP6LINMTU - The route is not currently active and the MTU was specified as \*IP6LINMTU, the MTU value of the line to which this route is bound.

**Change date.** The date of the most recent change to this route in the dynamic tables used by the TCP/IP<sub>v6</sub> protocol stack. It is returned as 8 characters in the form YYYYMMDD, where:

YYYY	Year
MM	Month
DD	Day

**Change time.** The time of the most recent change to this route in the dynamic tables used by the TCP/IP<sub>v6</sub> protocol stack. It is returned as 6 characters in the form HHMMSS, where:

HH	Hour
MM	Minute
SS	Second

**Configured route MTU.** A number representing the configured maximum transmission unit (MTU) value for this route in bytes.

The following is a special value:

0 \*IP6LINMTU - The route MTU was specified as \*IP6LINMTU, the MTU value of the line to which this route is bound.

**Duplicate indicator.** Indicates whether this route is a duplicate of another route in the routing table or not, and also whether there are any routes which are duplicates of this route. Use the Route status field to determine whether the route is in use or not.

Possible values are:

1	This route is not a duplicate of another route and it does not have any duplicates.
2	This route is not a duplicate of another route but it does have duplicates.
3	This route is a duplicate of another route.

**Expiration date.** The date when this route will expire or did expire. If the Expiration date and time are in the future, the route has not expired yet. If the Expiration date and time are in the past, then this route has expired and is still being returned for a short period of time to indicate that the route ceased to function because its lifetime expired. The Expiration date is returned as 8 characters in the form YYYYMMDD.

The meaning of the characters is as follows:

YYYY	Year
MM	Month
DD	Day

The following is a special value:

00000000 Infinite - this route has an infinite lifetime which never expires.

**Expiration time.** The time when this route will expire or has expired. If the Expiration date and time are in the future, the route has not expired yet. If the Expiration date and time are in the past, then this route

has expired and is still being returned for a short period of time to indicate that the route ceased to function because its lifetime expired. The Expiration time is returned as 6 characters in the form HHMMSS.

The meaning of the characters is as follows:

HH	Hour
MM	Minute
SS	Second

The following is a special value:

000000            Infinite - this route has an infinite lifetime which never expires.

**Is on-link.** Indicates whether this route is for a directly attached prefix (network) or not.

Possible values are:

0	Unknown, the on-link status of this route is undetermined.
1	Yes, this is a route to a directly attached prefix.

**Local binding line name.** The name of the communications line description to which this route is bound. This field is NULL padded.

The following are special values:

*LOOPBACK6	This route is bound to the loopback interface. Processing associated with the loopback interface does not extend to a physical line.
*TNLCFG64	This interface is bound to a configured 6-4 tunneling line.

**Local binding line status.** The current operational status of the communications line to which this route is bound.

Possible values are:

1	Active - The line is operational.
2	Inactive - The line is not operational.
3	Failed - The desired state of the line is Active, but it is currently in the Inactive state.

**Local binding line type.** Indicates the type of line to which this route is bound.

Possible values are:

-1	OTHER
-2	NONE - Line is not defined. This value is used for the following interfaces: *LOOPBACK6, *VIRTUALIP, *OPC. There is no line type value for these interfaces.
-3	ERROR - This value is displayed if any system errors other than those for *NOTFND are received while trying to determine the link type for an interface.
-4	NOTFND - Not found. This value is displayed if the line description object for this interface cannot be found.
1	ELAN - Ethernet local area network protocol.
2	TRLAN - Token-ring local area network protocol.
3	FR - Frame relay network protocol.
4	ASYN - Asynchronous communications protocol.
5	PPP - Point-to-point Protocol.
6	WLS - Wireless local area network protocol.

7	X.25 - X.25 protocol.
8	DDI - Distributed Data Interface protocol.
9	TDLC - Twinaxial Datalink Control. Used for TCP/IP over Twinax.
10	L2TP (Virtual PPP) - Layer Two Tunneling Protocol.
11	IPv6 Tunneling Line - Any kind of IPv6 over IPv4 tunnel.

**Next hop address family.** The address family of the Next Hop address for this route. Use this field to determine whether the IPv4 or IPv6 Next Hop field contains the value of the next hop.

Possible values are:

1	AF_INET - The next hop is an IPv4 address. Use the Next hop IPv4 fields.
2	AF_INET6 - The next hop is an IPv6 address. Use the Next hop IPv6 fields.

**Next hop IPv4.** The IPv4 internet address of the first system on the path from this system to the route destination in dotted-decimal format. The next hop will only be an IPv4 address when the route uses an IPv6 over IPv4 tunnel. This field is only valid when the value of the Next hop address family field is AF\_INET. This field is NULL padded

**Next hop IPv4 binary.** The binary representation of the Next hop IPv4 field. This field is only valid when the value of the Next hop address family field is AF\_INET.

**Next hop IPv6.** The IPv6 internet address of the first system on the path from your system to the route destination in IPv6 address format. This field is only valid when the value of the Next hop address family field is AF\_INET6. This field is NULL padded.

The following special value may be returned:

<i>*DIRECT</i>	This is the next hop value of a route that is automatically created. When an interface is added to this system, a route to the network the interface attaches to is also created.
----------------	---

**Next hop IPv6 binary.** The binary representation of the Next hop IPv6 field. Even though this field is defined as a character field, a binary IPv6 address is returned in it except when the following special character values are returned. This field is only valid when the value of the Next hop address family field is AF\_INET6.

The following special value may be returned:

<i>*DIRECT</i>	This is the next hop value of a route that is automatically created. When an interface is added to this system, a route to the network the interface attaches to is also created.
----------------	---

**Prefix length.** The prefix length defines how many bits of the route destination IPv6 address are in the prefix. It is a zoned decimal number which specifies how many of the left-most bits of the address make up the prefix. The prefix length is used to generate network and host addresses. This field is NULL padded.

**Prefix length binary.** Binary representation of the prefix length.

**Reserved.** An ignored field.

**Route destination.** The Internet Protocol version 6 (IPv6) address, in IPv6 address format, of the ultimate destination reached by this route. When used in combination with the prefix length the route destination identifies a route to a network or system. This field is NULL padded.

**Route destination binary.** The binary representation of the route destination. Even though this field is defined as a character field, a binary IPv6 address is returned in it.

**Route lifetime at creation.** The route lifetime value which this route had when it was first created, either automatically or by manual configuration. The route lifetime value is the length of time, in seconds, that a route remains in the route table. Only routes which are discovered on the network will have route lifetimes shorter than infinite. Valid values range from 1 through 4294967295 seconds.

The following is a special value:

0                      Infinite - this route had an infinite lifetime when it was created.

**Route lifetime remaining.** The length of time, in seconds, that a route remains in the route table. Only routes which are discovered on the network will have route lifetimes shorter than infinite. Valid values range from -31536000 through 4294967295 seconds. Negative values indicate that the route has expired, but it is being retained for a short period of time to show why the route ceased to function.

The following is a special value:

-1000000000            Infinite - this route has an infinite lifetime.

**Route source.** Specifies how this route was added to the IPv6 routing table.

The possible values are:

0                      Unknown  
1                      ICMPv6 Redirect - This route was added by the ICMPv6 redirect mechanism.  
2                      ICMPv6 Router Advertisement Router Lifetime - This route was added because of the presence of a non-zero value in the Router Lifetime field in a Router Advertisement packet received by the system.  
3                      ICMPv6 Router Advertisement Prefix Information Option - This route was added because of the presence of a Prefix Information Option on a Router Advertisement packet received by the system.  
4                      CFG RTE - This route was manually configured.  
5                      CFG IFC - This route was added when because of a manually configured interface.  
6                      Autoconfigured Interface - This route was added when because of an interface added by stateless autoconfiguration.  
7                      RIP - This route was added by the Routing Information Protocol (RIP).  
8                      OSPF - This route was added by the Open Shortest Path First (OSPF) routing protocol.  
9                      ROUTING - This route was determined to be necessary and added by the TCP/IP stack on this system.

**Route status.** The current state of the route.

Possible values are:

0                      Unknown  
1                      Active - This route is currently active and is in the current route search path.  
3                      Inactive - This route is not in the current route search path, and is not being used.

**Route type.** The type of route that this route is.

Possible values are:

0                      Unknown  
1                      DFTRROUTE - A default route.  
2                      DIRECT - A route to a network to which this system has a direct physical connection.  
3                      HOST - A route to a specific remote host.  
4                      NET - An indirect route to a remote network.

## Error Messages

Message ID	Error Message Text
TCP84C0 E	TCP/IP stack not active.
TCP84C5 E	API error providing TCP/IP Network Status list information.
TCP84C6 E	Internal operations error - RESULT &1 CC &2 RC &3 ERRNO &4.
TCP84C9 I	Information returned incomplete.
CPF0F03 E	Error in retrieving the user space that was created by the caller.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C1E E	Required parameter &1 omitted.
CPF3C21 E	Format name &1 is not valid.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	API contains a problem. See prior messages to determine why the failure occurred.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R1

[Top](#) | ["Communications APIs," on page 1](#) | [APIs by category](#)

---

## List Physical Interface ARP Table (QtocLstPhyIfcARPTbl) API

Required Parameter Group:	
<b>1</b>	Qualified user space name
<b>Input</b>	Char(20)
<b>2</b>	Format name
<b>Input</b>	Char(8)
<b>3</b>	Line name
<b>Input</b>	Char(10)
<b>4</b>	Error Code
<b>I/O</b>	Char(*)
Service Program: QTOCNETSTS	
Threadsafe: Yes	

The List Physical Interface ARP Table (QtocLstPhyIfcARPTbl) API returns a list of all entries in the ARP (Address Resolution Protocol) table for the specified line.

TCP/IP must be active on this system; otherwise, error message TCP84C0 will be issued.

## Authorities and Locks

*User Space Library Authority*  
\*EXECUTE

*User Space Authority*  
\*CHANGE

*User Space Lock*  
\*SHRNUP

## Required Parameter Group

**Qualified user space name**  
INPUT; CHAR(20)

The user space that receives the information, and the library in which it is located. The first 10 characters contain the user space name, and the second 10 characters contain the library name. You can use these special values for the library name:

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

**Format name**  
INPUT; CHAR(8)

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

**ARPT0100**

List of ARP table entries for a specified interface. Refer to "ARPT0100 Format" on page 152 for details on the format.

**Line name**  
INPUT; CHAR(10)

The name of the physical interface to retrieve ARP table entries for.

**Error Code**  
I/O; CHAR(\*)

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

## Format of ARP Table Lists

To request a list of ARP table entries for an interface, use format ARPT0100.

The ARP table list consists of:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:
  - ARPT0100 format.

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections.

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API Examples.

## Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name specified
28	1C	CHAR(10)	Line name specified

## Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name
10	A	CHAR(10)	User space library name used
20	14	CHAR(10)	Line name used

## ARPT0100 Format

The following information about an ARP table entry is returned for the ARPT0100 format. For detailed descriptions of the fields in the table, see “Field Descriptions.”

Offset		Type	Field
Dec	Hex		
0	0	CHAR(15)	Internet address
15	F	CHAR(1)	Reserved
16	10	BINARY(4)	Internet address binary
20	14	BINARY(4)	Line type
24	18	BINARY(4)	Ethernet type
28	1C	BINARY(4)	Type of entry
32	20	BINARY(4)	Data link connection identifier (DLCI)
36	24	BINARY(4)	Routing information field (RIF) valid mask
40	28	CHAR(18)	Routing information field (RIF)
58	3A	CHAR(17)	Physical address
75	4B	CHAR(1)	Reserved

## Field Descriptions

**Data link connection identifier (DLCI).** This field identifies a logical connection on a single physical Frame Relay link. Each logical connection has a unique integer identifying it. Valid values range from 1 to 255, and this field is only valid when the line type field corresponds to Frame Relay.

**Ethernet type.** The type of Ethernet framing in use. ONLY valid if the interface is using an ELAN (Ethernet) or WLS (Wireless) line.



-1	Both Ethernet Version 2 and IEEE 802.3 framing (only set for local or proxy entries)
1	Ethernet Version 2
6	IEEE 802.3

**Internet address.** The IP address of the interface in dotted-decimal notation.

**Internet address binary.** The binary representation of the IP address.

**Line type.** The type of physical line used by an interface. The possible values are:

1	ELAN - Ethernet local area network protocol.
2	TRLAN - Token-ring local area network protocol.
3	FR - Frame relay network protocol.
6	WLS - Wireless local area network protocol.
8	DDI - Distributed Data Interface protocol.

**Physical address.** The MAC address of the interface. Format: XX:XX:XX:XX:XX:XX, where 'X' is a hexadecimal digit.

**Reserved.** An ignored field.

**Routing information field (RIF).** The architected token-ring or FDDI source routing information. Use the RIF Valid Mask field to determine the validity of this field.

**Routing information field (RIF) valid mask.** Tells whether the RIF is valid for this ARP entry or not. The possible values are:

0	The RIF is not valid.
1	The RIF is valid.

**Type of entry.** The type of ARP table entry. The possible values are:

1	Dynamic - A normal ARP table entry which will be removed automatically after a period of inactivity.
2	Local - This interface is local to this host. Static entry.
3	Proxy - This interface is proxying ARP requests/replies for other machines. Static entry.

## Error Messages

TCP84C0 E	TCP/IP stack not active.
TCP84C3 E	The specified line name does not exist.
TCP84C4 E	The specified line name corresponds to a line type that does not support ARP.
TCP84C5 E	API error providing TCP/IP Network Status information.
TCP84C6 E	Internal operations error - RESULT &1 CC &2 RC &3 ERRNO &4.
TCP84C9 I	Information returned incomplete.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.

CPF9820 E            Not authorized to use library &1.  
 CPF9830 E            Cannot assign library &1.  
 CPF9872 E            Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R1

[Top](#) | ["Communications APIs," on page 1](#) | [APIs by category](#)

---

## List Physical Interface Data (QtocLstPhyIfcDta) API

Required Parameter Group:	
<b>1</b>	Qualified user space name
<b>Input</b>	Char(20)
<b>2</b>	Format name
<b>Input</b>	Char(8)
<b>3</b>	Error Code
<b>I/O</b>	Char(*)
	Service Program: QTOCNETSTS
Threadsafe: Yes	

The List Physical Interface Data (QtocLstPhyIfcDta) API returns a list of physical interfaces and detailed information about TCP/IP related data for each one. Depending on which output format is requested, IPv4 and also IPv6 information can be requested for each physical interface.

TCP/IP must be active on this system; otherwise error message TCP84C0 will be issued.

### Authorities and Locks

*User Space Library Authority*  
 \*EXECUTE

*User Space Authority*  
 \*CHANGE

*User Space Lock*  
 \*SHRNUP

### Required Parameter Group

**Qualified user space name**  
 INPUT; CHAR(20)

The user space that receives the information, and the library in which it is located. The first 10 characters contain the user space name, and the second 10 characters contain the library name. You can use these special values for the library name:

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

**Format name**  
 INPUT; CHAR(8)

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

<i>IFCD0100</i>	Basic physical interface data and detailed IPv4 specific data. Refer to “IFCD0100 Format” on page 156 for details on the format.
<i>IFCD0200</i>	Filter and IPSec Physical interface data. Refer to “IFCD0100 Format” on page 156 and “IFCD0200 Format” on page 159 for details on the format.
<i>IFCD0300</i>	Detailed IPv6 specific physical interface data. Refer to “IFCD0100 Format” on page 156 and “IFCD0300 Format” on page 162 for details on the format.

### Error Code

I/O; CHAR(\*)

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

## Format of Physical Interface Lists

To request a list of TCP/IP data for all physical interfaces, use format IFCD0100. For detailed information about Filter and IPSec physical interface data in addition to the IFCD0100 format data, use format IFCD0200.

The Physical Interface list consists of:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:
  - IFCD0100 format.
  - IFCD0200 format.
  - IFCD0300 format

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections.

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API Examples.

## Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name specified

## Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name
10	A	CHAR(10)	User space library name used

## Format of Returned Connection Data

To retrieve basic Physical interface data, and IPv4 specific statistics about each physical interface, use format “IFCD0100 Format.” To retrieve IPv4 Filter and IPSec statistics about each physical interface, in addition to format IFCD0100 information, use format “IFCD0200 Format” on page 159. To retrieve IPv6 specific information and statistics about each physical interface, in addition to format IFCD0100 information, use format “IFCD0300 Format” on page 162.

### IFCD0100 Format

The following data about a physical interface is returned for the IFCD0100 format. For detailed descriptions of the fields in the table, see “Field Descriptions.”

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Line type
4	4	BINARY(4)	Packet rules
8	8	BINARY(8)	Total bytes received
16	10	BINARY(8)	Total bytes sent
24	18	BINARY(4)	Total unicast packets received
28	1C	BINARY(4)	Total non-unicast packets received
32	20	BINARY(4)	Total inbound packets discarded
36	24	BINARY(4)	Total unicast packets sent
40	28	BINARY(4)	Total non-unicast packets sent
44	2C	BINARY(4)	Total outbound packets discarded
48	30	BINARY(4)	Physical interface status
52	34	CHAR(10)	Line description
62	3E	CHAR(17)	Physical address
79	4F	CHAR(8)	Date - retrieved
87	57	CHAR(6)	Time - retrieved
93	5D	CHAR(3)	Reserved
96	60	BINARY(4)	Offset to additional information
100	64	BINARY(4)	Length of additional information
104	68	BINARY(4)	Internet protocol version

### Field Descriptions

**Date - retrieved.** Date when information is retrieved and valid. Format: YYYYMMDD, where:

YYYY	Year
MM	Month
DD	Day

**Internet protocol version.** The version of the Internet Protocol (IP) that is currently in use on this line.

Possible values are:

1	IPv4
2	IPv6

**Length of additional information.** The length in bytes of additional information returned that is not part of format IFCD0100.

**Line description.** Each TCP/IP interface is associated with a physical network. This field displays the name of the communications line description that identifies the physical network associated with an interface.

The following special values may also be displayed:

* <i>IPI</i>	This interface is used by Internet Protocol (IP) over Internetwork Packet Exchange (IPX). A specific physical line is not associated with an interface used by IP over IPX (IPI).
	<b>Note:</b> As of V5R2, IP over IPX is no longer supported.
* <i>IPS</i>	The interface is used by Internet Protocol (IP) over SNA. A specific physical line is not associated with an interface used by IP over SNA (IPS).
* <i>LOOPBACK</i>	The interface is a loopback interface. Processing associated with a loopback interface does not extend to a physical line. There is no line description associated with a loopback address.
* <i>LOOPBACK6</i>	This line description is the IPv6 loopback line. Processing associated with a loopback line description does not extend to a physical line.
* <i>VIRTUALIP</i>	The virtual interface is a circuitless interface. It is used in conjunction with the associated local interface (LCLIFC) when adding standard interfaces. This special value is used to accommodate any of the following cases: <ol style="list-style-type: none"> <li>1. Load balancing. This is the means of having a fixed source IP address regardless of which interface the traffic is being distributed.</li> <li>2. Frame-relay multi-access network to define the local network IP address. This allows for multiple virtual circuits to share the same IP network.</li> <li>3. Alternate method of network access translation (NAT). This eliminates the need for a NAT box by assigning a globally unique single IP address directly to the box without the need to define an entire network.</li> <li>4. Unnumbered networks. This provides a means of associating a local source IP address for an unnumbered point-to-point network.</li> </ol>
* <i>OPC</i>	This special value is used if you are adding an OptiConnect interface over TCP/IP. This interface is attached to the optical bus (OptiConnect).
* <i>TNLCFG64</i>	This special value means this line description is a Configured 6-4 (IPv6 over IPv4) tunneling line. IPv6 Neighbor Discovery does not work over a Configured tunnel, so you don't get the benefit of stateless autoconfiguration.

**Line type.** Type of line used by an interface. The following link protocols are supported:

-1	OTHER -  <b>IPI - An Internet Protocol (IP) over Internetwork Pack Exchange (IPX).</b> <b>Note: As of V5R2, IP over IPX is no longer supported.</b> IPS - An Internet Protocol (IP) over SNA interface.  <b>WLS - Wireless local area network protocol.</b> TDLC - Twinaxial Datalink Control. Used for TCP/IP over Twinax.
-2	NONE - Line is not defined. There is no line type value for these interfaces.
-3	ERROR - This value is displayed if any system errors other than those for *NOTFND are received while trying to determine the link type for an interface.
-4	NOTFND - Not found. This value is displayed if the line description object for this interface cannot be found.
1	ELAN - Ethernet local area network protocol.
2	TRLAN - Token-ring local area network protocol.
3	FR - Frame relay network protocol.

4	ASYNC - Asynchronous communications protocol.
5	PPP - Point-to-point Protocol.
6	X.25 - X.25 protocol.
7	DDI - Distributed Data Interface.
8	OPC - OptiConnect interface.
9	LOOPBACK - Loopback interface.
10	IPv6 Tunneling Line - Any kind of IPv6 over IPv4 tunnel.

**Offset to additional information.** The offset in bytes to the rest of the information if a format other than IFCD0100 is requested.

**Packet rules.** Indicates what kind of packet rules data is available for a particular line.

0	None - No NAT and no filters are loaded for the line specified.
1	NAT - NAT is enabled for this line.
2	Filters - Filters are defined for this line.
3	NAT and Filters - NAT enabled and Filters defined.
4	Filters and IPSec - Filters and IPSec filters are defined for this line.
5	NAT and Filters and IPSec - NAT enabled and Filters and IPSec filters defined.

**Physical address.** The MAC address of the interface. Format: XX:XX:XX:XX:XX:XX, where 'X' is a hexadecimal digit.

**Physical interface status.** The current operational state of the physical interface (line).

0	Unknown - The status of this physical interface is unknown.
1	Active - The physical interface is operational.
2	Inactive - The physical interface is not operational.
3	Failed - The desired state of the physical interface is active, but it is currently in the inactive state.
4	Starting - The system is processing the request to start this physical interface.
5	Ending - The system is processing the request to start this physical interface.
6	Recovery Pending - An error has been detected with this physical interface and the system is recovering.
7	Recovery Canceled - An error has been detected with this physical interface and system recovery has been canceled.

**Reserved.** An ignored field.

**Time - retrieved.** Time when information is retrieved and valid. Format: HHMMSS, in 24 hour time, where:

HH	Hour
MM	Minute
SS	Second

**Total bytes received.** The total number of bytes received on the interface, including framing characters.

**Total bytes sent.** The total number of bytes transmitted out of the interface, including framing characters.

**Total inbound packets discarded.** The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space.

**Total non-unicast packets received.** The number of non-unicast (that is, broadcast or multicast) packets delivered to a higher-layer protocol.

**Total non-unicast packets sent.** The total number of packets that higher-level protocols requested be transmitted to a non-unicast (that is, broadcast or multicast) address, including those that were discarded or not sent.

**Total outbound packets discarded.** The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space.

**Total unicast packets received.** The number of unicast packets delivered to a higher-layer protocol.

**Total unicast packets sent.** The total number of packets that higher-level protocols requested to be transmitted to a unicast address, including those that were discarded or not sent.

## IFCD0200 Format

This format returns detailed Filter and IPSec Physical interface data in addition to data about a physical interface from the IFCD0100 format. For detailed descriptions of the fields in the table, see “Field Descriptions” on page 160.

Offset		Type	Field
Dec	Hex		
0	0		Returns everything from format IFCD0100.

Offset		Type	Field
Dec	Hex		
Decimal and hexadecimal offsets are reached by using the offset to additional information field in format IFCD0100.		CHAR(8)	Date - filter rules loaded or unloaded
		CHAR(6)	Time - filter rules loaded or unloaded
		CHAR(2)	Reserved
		BINARY(8)	Outbound filter packets discarded
		BINARY(8)	Outbound filter packets permitted
		BINARY(8)	Outbound packets non-filtered
		BINARY(8)	Outbound IPSec packets
		BINARY(8)	Outbound IPSec packets discarded - no connection
		BINARY(8)	Outbound IPSec packets discarded - ondemand
		BINARY(8)	Outbound IPSec packets discarded - VPN NAT
		BINARY(8)	Outbound IPSec packets discarded - other
		BINARY(8)	Outbound NAT packets
		BINARY(8)	Outbound NAT packets discarded
		BINARY(8)	Outbound packets discarded - other
		BINARY(8)	Outbound packets discarded - rule exception
		BINARY(8)	Inbound IPSec packets
		BINARY(8)	Inbound IPSec packets permitted
		BINARY(8)	Inbound IPSec packets discarded - no connection
		BINARY(8)	Inbound IPSec packets discarded - no AH/ESP
		BINARY(8)	Inbound IPSec packets discarded - ondemand
		BINARY(8)	Inbound IPSec packets discarded - VPN NAT
		BINARY(8)	Inbound IPSec packets discarded - anti-replay fail
		BINARY(8)	Inbound IPSec packets discarded - selector mismatch
		BINARY(8)	Inbound IPSec packets discarded - mode mismatch
		BINARY(8)	Inbound IPSec packets discarded - authentication error
		BINARY(8)	Inbound IPSec packets discarded - other
		BINARY(8)	Inbound NAT packets
		BINARY(8)	Inbound filter packets discarded
		BINARY(8)	Inbound filter packets permitted
		BINARY(8)	Inbound packets non-filtered
		BINARY(8)	Inbound packets discarded - other
		BINARY(8)	Inbound packets discarded - rule exception
		BINARY(4)	NAT rules
	BINARY(4)	Filter rules	
	BINARY(4)	IPSec rules	

## Field Descriptions

### Date - filter rules loaded or unloaded.

Date when the filter rules were most recently successfully loaded on or unloaded from this interface.  
Format: YYYYMMDD, where:

YYYY                    Year  
MM                       Month  
DD                       Day



The following is a special value:

00000000 Rules have never been loaded since interface was loaded.

**Filter rules.** Indicates whether filter rules exist on the system. The possible values are:

0	No filter rules exist.
1	Filter rules exist.

**Inbound IPSec packets.** Total inbound IPSec packets (AH or ESP) processed without error.

**Inbound IPSec packets permitted.** Total inbound packets permitted by pre-IPSec filters.

**Inbound IPSec packets discarded - authentication error.** Authentication error or failed.

**Inbound IPSec packets discarded - no connection.** Total inbound packets discarded because a VPN connection was not started.

**Inbound IPSec packets discarded - no AH/ESP.** Total inbound packets discarded because packet should have had a AH or ESP header, and did not.

**Inbound IPSec packets discarded - ondemand.** Total inbound packets discarded due to a starting on-demand VPN connection.

**Inbound IPSec packets discarded - anti-replay fail.** Total inbound packets discarded due to failed anti-replay audit.

**Inbound IPSec packets discarded - mode mismatch.** Total inbound packets discarded because the mode (tunnel or transport) of the packet did not match the mode of the VPN connection.

**Inbound IPSec packets discarded - other.** Total inbound packets discarded for other reasons, relating to IPSec.

**Inbound IPSec packets discarded - selector mismatch.** Total inbound packets discarded because the packet did not match the VPN connection (selectors).

**Inbound IPSec packets discarded - VPN NAT.** Total inbound packets that could not be NAT'd because an IP address was not available from a VPN NAT pool.

**Inbound NAT packets.** Total inbound packets processed by conventional NAT.

**Inbound filter packets discarded.** Total inbound packets discarded by filter action = DENY.

**Inbound filter packets permitted.** Total inbound packets permitted by filter action = PERMIT.

**Inbound packets non-filtered.** Total inbound packets not filtered (occurs only when no filters exist).

**Inbound packets discarded - other.** Total inbound packets discarded for some other reason.

**Inbound packets discarded - rule exception.** Total inbound packets discarded for exception reason.

**IPSec rules.** Indicates whether IPSec filter rules exist on the system. The possible values are:

0 No IPSec filter rules exist.  
1 IPSec filter rules exist.

**NAT rules.** Indicates whether NAT rules exist on the system. The possible values are:

0 No NAT rules exist.  
1 NAT rules exist.

**Outbound filter packets discarded.** Total outbound packets discarded by filter action = DENY.

**Outbound filter packets permitted.** Total outbound packets permitted by filter action = PERMIT.

**Outbound packets non-filtered.** Total outbound packets not filtered (occurs only when no filters exist).

**Outbound IPSec packets.** Total outbound IPSec packets (AH or ESP) processed without error.

**Outbound IPSec packets discarded - no connection.** Total outbound packets that could not be handled by IPSec because a VPN connection was not started.

**Outbound IPSec packets discarded - ondemand.** Total outbound packets discarded due to a starting on-demand VPN connection.

**Outbound IPSec packets discarded - other.** Total outbound packets that could not be handled for other reasons.

**Outbound IPSec packets discarded - VPN NAT.** Total outbound packets that could not be NAT'd because an IP address was not available from a VPN NAT pool.

**Outbound NAT packets.** Total outbound packets processed by conventional NAT.

**Outbound NAT packets discarded.** Total outbound packets that could not be handled by masquerade NAT due to lack of available conversation.

**Outbound packets discarded - other.** Total outbound packets discarded for some other reason.

**Outbound packets discarded - rule exception.** Total outbound packets discarded for exception reason.

**Reserved.** An ignored field.

**Time - filter rules loaded or unloaded.** Time when the filter rules were most recently successfully loaded on or unloaded from this interface. Format: HHMMSS, in 24 hour time, where:

HH	Hour
MM	Minute
SS	Second

The following is a special value:

000000 Rules have never been loaded since interface was loaded.

## IFCD0300 Format

This format returns detailed IPv6 specific information and statistics for each Physical interface, in addition to data about a physical interface from the IFCD0100 format. For detailed descriptions of the fields in the table, see "Field Descriptions" on page 163.

Offset		Type	Field
Dec	Hex		
0	0		Returns everything from format IFCD0100.
Decimal and hexadecimal offsets are reached by using the offset to additional information field in format IFCD0100.		BINARY(4)	Packet rules - IPv6
		BINARY(8)	Total IPv6 bytes received
		BINARY(8)	Total IPv6 bytes sent
		BINARY(4)	Total IPv6 unicast packets received
		BINARY(4)	Total IPv6 multiicast packets received
		BINARY(4)	Total IPv6 anycast packets received
		BINARY(4)	Total inbound IPv6 packets discarded
		BINARY(4)	Total IPv6 unicast packets sent
		BINARY(4)	Total IPv6 multicast packets sent
		BINARY(4)	Total IPv6 anycast packets sent
		BINARY(4)	Total outbound IPv6 packets discarded
		CHAR(25)	IPv6 interface identifier
		CHAR(7)	Reserved
		BINARY(8)	IPv6 interface identifier binary
		BINARY(4)	MTU - configured
		BINARY(4)	MTU - current
		BINARY(4)	Hop limit - configured
		BINARY(4)	Hop limit - current
		BINARY(4)	Use stateless autoconfig
		BINARY(4)	Use stateful address configuration
		BINARY(4)	Use other stateful configuration
		BINARY(4)	Accept router advertisements
		BINARY(4)	Accept redirects
		BINARY(4)	Neighbor discovery base reachable time - configured
		BINARY(4)	Neighbor discovery base reachable time - current
		BINARY(4)	Neighbor discovery reachable time
		BINARY(4)	Neighbor solicitation retransmit interval - configured
		BINARY(4)	Neighbor solicitation retransmit interval - current
		BINARY(4)	Duplicate address detection max transmits
		CHAR(15)	Local tunnel endpoint IPv4 address
		CHAR(1)	Reserved
		BINARY(4)	Local tunnel endpoint IPv4 address binary

## Field Descriptions

**Accept redirects.** Whether the system is currently accepting and using ICMPv6 Redirects that it receives on this physical interface.

Possible values are:

- 0 No - this interface is not accepting redirects.
- 1 Yes - this interface is accepting redirects.

**Accept router advertisements.** Whether the system is currently accepting and using Router Advertisements that it receives on this physical interface.

Possible values are:

- 0 No - this interface is not accepting router advertisements.
- 1 Yes - this interface is accepting router advertisements.

**Duplicate address detection max transmits.** The maximum number of consecutive Neighbor Solicitation messages which will be sent using this physical interface when TCP/IPv6 performs Duplicate Address Detection (DAD) on a tentative address.

The following special value may be returned:

- 0 This physical interface is currently configured to not perform Duplicate Address Detection.

**Hop limit - configured.** The configured IPv6 Hop Limit value specified for this physical interface. The Hop limit field is the IPv6 replacement for the IPv4 Time to live (TTL) field. The Hop limit value specifies a relative limit on the number of hops across which an IPv6 datagram remains active. The Hop limit value is hop count that is decremented by each gateway to prevent internet routing loops. The default Hop limit value is 64. Valid values range from 1 through 255 hops.

**Hop limit - current.** The IPv6 Hop Limit value currently in effect for this physical interface. The Hop Limit field is the IPv6 replacement for the IPv4 Time to live (TTL) field. The Hop Limit value specifies a relative limit on the number of hops across which an IPv6 datagram remains active. The Hop Limit value is hop count that is decremented by each gateway to prevent internet routing loops. The default Hop Limit value is 64. If the current Hop Limit value differs from the configured Hop Limit value, then it has been set by a Hop Limit value received in a Router Advertisement packet. Valid values range from 1 through 255 hops.

**IPv6 interface identifier.** A 64-bit number which is combined with prefixes to create complete IPv6 addresses for the physical interface. By default it is based on the link layer (MAC) address, if one exists. The interface identifier is represented here in standard IPv6 address format notation. It does not include a leading "::" for the first 64 bits of a full IPv6 address, and it may include an embedded IPv4 address at the end. This field is NULL padded.

**IPv6 interface identifier binary.** Binary representation of the IPv6 interface identifier.

**Local tunnel endpoint IPv4 address.** The IPv4 address of the local tunnel endpoint of this tunnel, returned in dotted decimal format. This field is NULL padded.

The following special value may be returned:

- 0.0.0.0 This physical interface is not a tunnel, so this field does not apply.

**Local tunnel endpoint IPv4 address binary.** Binary representation of the Local tunnel endpoint IPv4 address.

The following special value may be returned:

- 0 This physical interface is not a tunnel, so this field does not apply.

**MTU - configured.** The configured maximum transmission unit (MTU) value specified for this physical interface.

The following is a special value:

0 LIND - The MTU was configured as \*LIND, the MTU value from the line description.

**MTU - current.** Maximum transmission unit (MTU) value currently in effect for this physical interface.

The following is a special value:

0 LIND - The interface is not active currently and the MTU was configured as \*LIND.

**Neighbor discovery base reachable time - configured.** The configured Neighbor Discovery (ND) Base Reachable Time value, in seconds, specified for this physical interface. The ND Base Reachable Time value is a base time value used for computing the random ND Reachable Time value. The default ND Base Reachable Time is 30 seconds. Valid values range from 10 through 100 seconds.

**Neighbor discovery base reachable time - current.** The Neighbor Discovery (ND) Base Reachable Time value, in seconds, currently in effect for this physical interface. The ND Base Reachable Time value is a base time value used for computing the random ND Reachable Time value. The default ND Base Reachable Time is 30 seconds. If the current ND Base Reachable Time value differs from the configured value, then it has been set by a ND Base Reachable Time value received in a Router Advertisement packet. Valid values range from 10 through 100 seconds.

**Neighbor discovery reachable time.** The current Neighbor Discovery (ND) Reachable Time value, in seconds, for this physical interface. The ND Reachable Time value is the amount of time, in seconds, that a neighbor is considered reachable after receiving a reachability confirmation. The ND Reachable Time value is randomly calculated, using the ND Base Reachable Time and a couple constants. This calculation is performed to prevent Neighbor Unreachability Detection (NUD) messages from synchronizing with each other.

**Neighbor solicitation retransmit interval - configured.** The configured Neighbor Solicitation (NS) Retransmit Interval value, in seconds, specified for this physical interface. The NS Retransmit Interval is the time, in seconds, between retransmissions of Neighbor Solicitation messages to a neighbor when resolving the link-layer address, or when probing the reachability of a neighbor. The default NS retransmit interval is 1 second. Valid values range from 1 through 10 seconds.

**Neighbor solicitation retransmit interval - current.** The Neighbor Solicitation (NS) Retransmit Interval value currently in effect for this physical interface. The NS Retransmit Interval is the time, in seconds, between retransmissions of Neighbor Solicitation messages to a neighbor when resolving the link-layer address, or when probing the reachability of a neighbor. The default NS Retransmit Interval is 1 second. If the current NS Retransmit Interval value differs from the configured value, then it has been set by a NS Retransmit Interval value received in a Router Advertisement packet. Valid values range from 1 through 10 seconds.

**Packet rules - IPv6.** Indicates what kind of IPv6 packet rules are loaded on a particular line.

Possible values are:

-1	Other - An unknown Packet rules value.
0	None - No NAT and no filters are loaded for the line specified.
1	NAT - NAT is enabled for this line.
2	Filters - Filters are defined for this line.
3	NAT and Filters - NAT enabled and Filters defined for this line.
4	Filters and IPSec - Filters and IPSec filters are defined for this line.
5	NAT and Filters and IPSec - NAT enabled and Filters and IPSec filters defined for this line.

**Reserved.** An ignored field.

**Total inbound IPv6 packets discarded.** The number of inbound IPv6 packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space.

**Total IPv6 anycast packets received.** The number of IPv6 anycast packets delivered to a higher-layer protocol.

**Total IPv6 anycast packets sent.** The number of IPv6 anycast packets that higher-level protocols requested be transmitted, including those that were discarded.

**Total IPv6 bytes received.** The total number of IPv6 bytes received on the interface, including framing characters.

**Total IPv6 bytes sent.** The total number of IPv6 bytes transmitted out of the interface, including framing characters.

**Total IPv6 multicast packets received.** The number of IPv6 multicast packets delivered to a higher-layer protocol.

**Total IPv6 multicast packets sent.** The number of IPv6 multicast packets that higher-level protocols requested be transmitted, including those that were discarded.

**Total IPv6 unicast packets received.** The number of IPv6 unicast packets delivered to a higher-layer protocol.

**Total IPv6 unicast packets sent.** The number of IPv6 unicast packets that higher-level protocols requested be transmitted, including those that were discarded.

**Total outbound IPv6 packets discarded.** The number of outbound IPv6 packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space.

**Use other stateful configuration.** Whether the TCP/IPv6 stack has been informed by a Router Advertisement to use non-address stateful (that is, DHCPv6) configuration information that it receives on this physical interface.

Possible values are:

- 1 UNKNOWN - The system has not received any Router Advertisements on this physical interface.
- 0 NO - The system has been informed to not use any non-address stateful configuration information that it receives on this physical interface.
- 1 YES - The system has been informed to use any non-address stateful configuration information that it receives on this physical interface.

**Use stateful address configuration.** Whether the TCP/IPv6 stack has been informed by a Router Advertisement to use stateful (that is, DHCPv6) configuration information that it receives on this physical interface for the purpose of address autoconfiguration.

Possible values are:

- 1 UNKNOWN - The system has not received any Router Advertisements on this physical interface.
- 0 NO - The system has been informed to not use stateful configuration information that it receives on this physical interface for the purpose of address autoconfiguration.
- 1 YES - The system has been informed to use stateful configuration information that it receives on this physical interface for the purpose of address autoconfiguration.

Use **stateless autoconfig**. Whether the TCP/IPv6 stack performs stateless autoconfiguration on this physical interface or not.

Possible values are:

- 0 NO - The system will not perform the stateless autoconfig algorithms on this physical interface.
- 1 YES - The system will perform the stateless autoconfig algorithms on this physical interface.

## Error Messages

Message ID	Error Message Text
TCP84C0 E	TCP/IP stack not active.
TCP84C5 E	API error providing TCP/IP Network Status list information.
TCP84C6 E	Internal operations error - RESULT &1 CC &2 RC &3 ERRNO &4.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9801 E	Object &2 in library &3 not found.
CPF9802 E	Not authorized to object &2 in &3.
CPF9803 E	Cannot allocate object &2 in library &3.
CPF9807 E	One or more libraries in library list deleted.
CPF9808 E	Cannot allocate one or more libraries on library list.
CPF9810 E	Library &1 not found.
CPF9820 E	Not authorized to use library &1.
CPF9830 E	Cannot assign library &1.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R1

[Top](#) | ["Communications APIs," on page 1](#) | [APIs by category](#)

---

## List PPP Connection Profiles (QtocLstPPPCnnPrf) API

Required Parameter Group:	
1	Qualified user space name
<b>Input</b>	Char(20)
2	Format name
<b>Input</b>	Char(8)
3	Error Code
<b>I/O</b>	Char(*)
	Service Program: QTOCPPPAPI
Threadsafe: Yes	

The List PPP Connection Profiles API (QtocLstPPPCnnPrf) returns a list of PPP connection profiles with some basic information about each profile.

## Authorities and Locks

*User Space Library Authority*  
\*EXECUTE

*User Space Authority*  
\*CHANGE

*User Space Lock*  
\*SHRNUP

## Required Parameter Group

**Qualified user space name**  
INPUT; CHAR(20)

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

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

**Format name**  
INPUT; CHAR(8)

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

*PRFD0100*    Connection profile lists. Refer to "PRFD0100 Format" on page 169 for details on the format.

**Error code**  
I/O; CHAR(\*)

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

## Format of Connection Profile Lists

To request a list of PPP Connection Profiles, use format PRFD0100.

The PPP Connection Profile list consists of:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:
  - PRFD0100 format

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections.

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API Examples.



## Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name specified

## Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name
10	A	CHAR(10)	User space library name used

## PRFD0100 Format

The following data about a PPP Connection Profile is returned for the PRFD0100 format. For detailed descriptions of the fields in the table, see “Field Descriptions.”

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Profile mode
4	4	BINARY(4)	Connection protocol
8	8	BINARY(4)	Connection status
12	C	BINARY(4)	Connection type
16	10	BINARY(4)	Profile job type
20	14	BINARY(4)	Multilink connection enabled
24	18	CHAR(10)	Profile name
34	22	CHAR(10)	Line name
44	2C	CHAR(10)	Line type
54	36	CHAR(10)	Job name
64	40	CHAR(10)	Job user profile
74	4A	CHAR(6)	Job number
80	50	CHAR(50)	profile description
130	82	CHAR(10)	Dial-on-demand peer answer profile
140	8C	» BINARY(4)	Automatic start
144	90	CHAR(16)	Reserved «

## Field Descriptions



**Automatic start.** Whether the profile is started automatically when the TCP/IP stack is activated. Possible values are:

0 NO. This profile is not started automatically.

1 YES. This profile is started automatically.



**Connection protocol.** The type of point-to-point connection provided by the profile job.

1 SLIP.  
2 PPP.

**Connection status.** The current connection of job status of the profile job. Values are as follows:

1	Inactive
2	Session error
3	Ended - information available
4	Session start submitted
11	Session job starting
12	Session job ending
13	Session ended - job log pending
14	Adding TCP/IP configuration
15	Removing TCP/IP configuration
16	Message pending
17	Session error
18	Starting TCP/IP
19	Ending TCP/IP
21	Calling remote system
22	Waiting for incoming call
23	Connecting
24	Active
26	Switched line-dial on demand
27	Waiting for incoming call - switched line-answer enabled dial on demand
28	Waiting for shared line resource
29	Requesting shared line resource
31	LCP initializing
32	LCP starting
33	LCP closing
34	LCP closed
35	LCP waiting for configuration request
36	LCP configuring
37	LCP authenticating
41	IPCP initializing
42	IPCP starting
43	IPCP ending
44	IPCP stopped
45	IPCP waiting for configuration request
46	IPCP configuring
47	IPCP opening
51	Multi-connection - waiting for incoming call(s)
52	Multi-connection L2TP initiator waiting for tunnel
53	Multi-connection - at least one connection active
54	Multi-hop terminator starting multi-hop initiator
55	Multi-hop initiator establishing second hop tunnel
56	Multi-hop initiator tunnel pre-started
57	Multi-hop connection active
58	Starting VPN connection
59	Negotiating IPSEC SA
60	PPPoE discovery stage
61	PPPoE session stage

**Connection type.** The type of connection provided by the profile job. Values are:

1	Switched or dialed connection
2	Leased or non-switched connection
3	Virtual circuit connection
4	PPPoE

**Dial-on-demand peer answer profile.** Specifies the name of the answer only profile that answers incoming calls from the remote peer.

**Job name.** The job name of the job that currently or most recently executed this profile job description. This field is blank if this connection profile job has not been run.

**Job number.** The job number of the job that currently or most recently executed this profile job description. This field is blank if this connection profile job has not been run.

**Job user profile.** The user profile of the job that currently or most recently executed this profile job description. This field is blank if this connection profile job has not been run.

**Line name.** Each TCP/IP interface is associated with a physical network. This field displays the name of the communications line description that identifies the physical network associated with an interface. May be blank when Line type selection is \*LINEPOOL and no member line has been selected.

**Line type.** The type of line connection defined in this connection profile. Possible values are:

*PPP	PPP line description
*LINEPOOL	Line name is a member of a line pool
*L2TP	L2TP line description
*PPPOE	PPPoE line description
*ERROR	The selected line type is undefined or is improperly defined

**Multilink connection enabled.** Whether multilink connections are enabled for the profile. Values are:

0	No
1	Yes

**Profile description.** The text description of the function performed by this profile connection job.

**Profile job type.** The type of job support required for the profile.

1	Single connection profile
2	Multi-connection or multilink connection profile

**Profile mode.** The function provided by the profile job. Values are:

1	Dial only.
2	Answer only.
3	Dial-on-demand.
4	Answer enabled dial-on-demand.
5	L2TP virtual Initiator.
6	Remote peer enabled dial-on-demand.
7	L2TP initiator-on-demand.
8	L2TP multihop initiator.
9	PPPoE initiator.

**Profile name.** The name of this connection profile description.

**Reserved.** An ignored field.

## Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3C21 E	Format name &1 is not valid.
CPF3CAA E	List is too large for user space &1.
CPF3CF1 E	Error code parameter not valid.
CPF811A E	User space &4 in &9 damaged.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R1

[Top](#) | [“Communications APIs,”](#) on page 1 | [APIs by category](#)

---

## List TCP/IP Point-to-Point Jobs (QTOCLPPJ) API

Required Parameter Group:	
<b>1</b>	Qualified user space name
<b>Input</b>	Char(20)
<b>2</b>	Format name
<b>Input</b>	Char(8)
<b>3</b>	Point-to-point connection profile name
<b>Input</b>	Char(10)
<b>4</b>	Error code
<b>I/O</b>	Char(*)
	Threadsafe: Yes

The List TCP/IP Point-to-Point Jobs (QTOCLPPJ) API returns information about each connection job currently associated with the specified point-to-point connection profile.

## Authorities and Locks

*User Space Authority*

\*CHANGE

*Authority to Library Containing User Space*

\*EXECUTE

## Required Parameter Group

**Qualified user space name**  
INPUT; CHAR(20)

The user space that receives the information and the library in which it is located. The first 10 characters contain the user space name. The second 10 characters contain the library name. You can use these special values for the library name:

\*CURLIB                      The job's current library

\*LIBL

The library list

### Format name

INPUT; CHAR(8)

The content and format of the list returned. The possible format name is:

PPPJ0100

Each entry in the list contains information about a point-to-point job associated with the specified point-to-point connection profile name. The specified profile must be active for job information to be returned. If the specified profile is not active, an empty list will be returned.

See “Format of Point-to-Point Jobs List” for a description of the format.

### Point-to-point connection profile name

INPUT; CHAR(10)

The name of the point-to-point connection profile for which connection job information is being requested.

### Error code

I/O; CHAR(\*)

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

## Format of Point-to-Point Jobs List

The point-to-point jobs list consists of:

- A user area
- A generic header
- An input parameter section
- A header section
- A list data section:

- PPPJ0100 format

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections. For detailed descriptions of the fields in the list returned, see “Field Descriptions” on page 174.

When you retrieve list entry information from a user space, you must use the entry size returned in the generic header. The size of each entry may be padded at the end. If you do not use the entry size, the result may not be valid. For examples of how to process lists, see API examples.

## Input Parameter Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name specified
10	A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name specified
28	1C	CHAR(10)	Point-to-point connection profile name specified

## Header Section

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	User space name used
10	A	CHAR(10)	User space library name used
20	14	CHAR(10)	Point-to-point connection profile name

## PPPJ0100 Format

Offset		Type	Field
Dec	Hex		
0	0	CHAR(10)	Job name
10	A	CHAR(10)	Job user name
20	14	CHAR(6)	Job number
26	1A	CHAR(16)	Internal job identifier
42	2A	CHAR(10)	Line name
52	34	BINARY(4)	Connection status
56	38	CHAR(15)	Local IP address
71	47	CHAR(15)	Remote IP address
86	56	CHAR(48)	Connected user name

## Field Descriptions

**Connection status.** The current status of the connection.

**Connected user name.** The name of the user who initiated the point-to-point connection. The user name is available only if authentication is enabled for the point-to-point connection profile; otherwise, \*NONE will be returned.

**Format name.** The name of the format used to list the point-to-point connection jobs associated with an active point-to-point connection profile.

**Internal job identifier.** A value sent to other APIs to speed the process of locating the job on the system. The identifier is not valid following an initial program load (IPL). If you attempt to use it after an IPL, an exception occurs.

**Job name.** The simple job name of the point-to-point job.

**Job number.** The system-assigned job number of the point-to-point job.

**Job user name.** The user name under which the point-to-point job is running. This will be defined as QTCP for point-to-point connection profiles.

**Line name.** The name of the line associated with the point-to-point connection.

**Local IP address.** The IP address assigned to the local end of the point-to-point connection. The IP address is in dotted decimal format.

**Point-to-point connection profile name.** The name of the point-to-point connection profile for which connection job information is being requested.

**Remote IP address.** The IP address assigned to the remote end of the point-to-point connection. The IP address is in dotted decimal format.

**User space library name.** The name of the library containing the user space.

**User space name.** The user space used to return the list of point-to-point connection jobs associated with an active point-to-point connection profile.

## Error Messages

Message ID	Error Message Text
CPF24B4 E	Severe error while addressing parameter list.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF3C21 E	Format name &1 is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.
TCP8211 E	Point-to-point profile &1 not found.

API introduced: V4R4

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

---

## Remove ARP Table Entry (QtocRmvARPTbIE) API

Required Parameter Group:	
<b>1</b>	Line name
<b>Input</b>	Char(10)
<b>2</b>	Internet address
<b>Input</b>	Binary(4)
<b>3</b>	Entry type
<b>Input</b>	Char(10)
<b>4</b>	Error code
<b>I/O</b>	Char(*)
	Service Program: QTOCNETSTS
	Threadsafe: Yes

The Remove ARP Table Entry (QtocRmvARPTbIE) API removes one or all dynamic entries from the ARP (Address Resolution Protocol) table for the specified line. Local interface entries cannot be removed.

TCP/IP must be active on this system; otherwise, error message TCP84C0 is issued.

## Authorities and Locks

*Special Authority*  
\*IOSYSCFG

## Required Parameter Group

### Line name

INPUT; CHAR(10)

The name of the physical interface corresponding to the ARP table from which to remove entries.

### Internet address

INPUT; BINARY(4)

The IP address of the entry to remove from the ARP table. This must be 0 when trying to remove all dynamic ARP entries.

### Entry type

INPUT; CHAR(10)

Whether a single entry or all entries are removed from the ARP table. The possible types are:

*\*IPADDR*

The Internet address field corresponds to a single entry to be removed.

*\*ALL*

The Internet address field must be 0 and all ARP table entries will be removed.

### Error code

I/O; CHAR(\*)

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

## Error Messages

Message ID	Error Message Text
TCP84C0 E	TCP/IP stack not active.
TCP84C1 E	The specified Internet address was not found in the ARP table.
TCP84C2 E	ARP entry is local and cannot be deleted.
TCP84C3 E	The specified line name does not exist.
TCP84C4 E	The specified line name corresponds to a line type that does not support ARP.
TCP84C6 E	Internal operations error.
TCP84C8 E	ARP API parameter not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R1

[Top](#) | ["Communications APIs,"](#) on page 1 | [APIs by category](#)



## Retrieve Network Connection Data (QtocRtvNetCnnDta) API

Required Parameter Group:	
1	Receiver variable
<b>Output</b>	Char(*)
2	Length of receiver variable
<b>Input</b>	Binary(4)
3	Format name
<b>Input</b>	Char(8)
4	Socket connection request
<b>Input</b>	Char(*)
5	Error Code
<b>I/O</b>	Char(*)
Service Program: QTOCNETSTS	
Threadsafe: Yes	

The Retrieve Network Connection Data (QtocRtvNetCnnDta) API retrieves detailed information about a specified IPv4 or IPv6 network connection - including jobs using the connection. It also retrieves information about IPv4 and IPv6 connection totals.

TCP/IP must be active on this system, otherwise TCP84C0 message will be issued.

### Authorities and Locks

None.

### Required Parameter Group

#### Receiver variable

OUTPUT; CHAR(\*)

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

#### Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable. If this value is larger than the actual size of the receiver variable, the result may not be predictable. The minimum length is 8 bytes.

#### Format name

INPUT; CHAR(8)

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

<i>NCND0100</i>	TCP/IPv4 connection totals. Refer to "NCND0100 Format" on page 179 for details on the format.
<i>NCND0200</i>	Detailed TCP or UDP connection status for a specific IPv4 socket connection in addition to TCP/IPv4 connection totals. Refer to "NCND0100 Format" on page 179 and "NCND0200 Format" on page 181 for details on the format.
<i>NCND1100</i>	TCP/IPv6 connection totals. Refer to "NCND1100 Format" on page 187 for details on the format.

Detailed TCP or UDP connection status for a specific IPv6 socket connection in addition to TCP/IPv6 connection totals. Refer to “NCND1100 Format” on page 187 and “NCND1200 Format” on page 189 for details on the format.

### Socket connection request

INPUT; CHAR(\*)

The protocol, local address, local port, remote address and remote port identify the connection for which information is to be retrieved. This parameter is ignored when format NCND0100 or format NCND1100 is requested. Refer to “Socket Connection Request Format” for details on the format.

### Error code

I/O; CHAR(\*)

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

## Socket Connection Request Format

Information passed in the socket connection request parameter must be in one of the following two formats. The first format is for IPv4 connections, and the second is for IPv6 connections. The value of the Protocol field determines the format of the rest of the Socket Connection Request. For detailed descriptions of the fields in the table, see “Field Descriptions.”

### IPv4 connection (Protocol field value is 1 or 2)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Protocol
4	4	BINARY(4)	Local IPv4 address
8	8	BINARY(4)	Local port number
12	C	BINARY(4)	Remote IPv4 address
16	10	BINARY(4)	Remote port number

### IPv6 connection (Protocol field value is 3 or 4)

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Protocol
4	4	CHAR(16)	Local IPv6 address
20	14	CHAR(4)	Local port number
24	18	CHAR(16)	Remote IPv6 address
40	28	BINARY(4)	Remote port number

## Field Descriptions

**Local IPv4 address.** The IPv4 address of the host at the local end of the connection.

**Local IPv6 address.** The IPv6 address of the host at the local end of the connection. Even though this field is defined as a character field, it must be stored in binary. It is recommended that you use the Sockets `in6_addr` structure.

**Local port number.** The port number of the local end of the connection.

**Protocol.** The type and IP version of connection protocol.

Possible values are:

- 0 TCP/IP connection totals when using format NCND0100 or format NCND1100.
- 1 TCP/IPv4 - A Transmission Control Protocol (TCP) over IPv4 connection or socket request.
- 2 UDP/IPv4 - A User Datagram Protocol (UDP) over IPv4 socket request.
- 3 TCP/IPv6 - A Transmission Control Protocol (TCP) over IPv6 connection or socket request.
- 4 UDP/IPv6 - A User Datagram Protocol (UDP) over IPv6 socket request.

**Remote IPv4 address.** The IPv4 address of the host at the remote end of the connection.

**Remote IPv6 address.** The IPv6 address of the host at the remote end of the connection. Even though this field is defined as a character field, it must be stored in binary. It is recommended that you use the Sockets `in6_addr` structure.

**Remote port number.** The port number of the remote end of the connection.

## Format of Returned Connection Data

To retrieve the current TCP/IPv4 connection totals, use format “NCND0100 Format.”

To retrieve the current TCP/IPv6 connection totals, use format “NCND1100 Format” on page 187.

For detailed TCP and UDP connection status for a specific IPv4 socket connection in addition to the TCP/IPv4 connection totals, use format “NCND0200 Format” on page 181.

For detailed TCP and UDP connection status for a specific IPv6 socket connection in addition to the TCP/IPv6 connection totals, use format “NCND1200 Format” on page 189.

## NCND0100 Format

Format NCND0100 returns information regarding the TCP/IPv4 connection totals. For detailed descriptions of the fields in the table, see “Field Descriptions” on page 180.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	TCP connections currently established
12	C	BINARY(4)	TCP active opens
16	10	BINARY(4)	TCP passive opens
20	14	BINARY(4)	TCP attempted opens that failed
24	18	BINARY(4)	TCP established and then reset
28	1C	BINARY(4)	TCP segments sent
32	20	BINARY(4)	TCP retransmitted segments
36	24	BINARY(4)	TCP reset segments
40	28	BINARY(4)	TCP segments received
44	2C	BINARY(4)	TCP segments received in error
48	30	BINARY(4)	UDP datagrams sent
52	34	BINARY(4)	UDP datagrams received
56	38	BINARY(4)	UDP datagrams not delivered application port not found

Offset		Type	Field
Dec	Hex		
60	3C	BINARY(4)	UDP datagrams not delivered other datagrams in error
64	40	BINARY(4)	Offset to additional information
68	44	BINARY(4)	Length of additional information

## Field Descriptions

**Bytes available.** All of the available bytes for use in your application.

**Bytes returned.** The number of bytes returned to the user. This may be some but not all the bytes available.

**Length of Additional Information.** The length in bytes of additional information returned that is not part of format NCND0100.

**Offset to Additional Information.** The offset in bytes to the rest of the information if a format other than NCND0100 is requested.

**TCP active opens.** The number of times TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state. This number is an indication of the number of times this local system opened a connection to a remote system.

**TCP attempted opens that failed.** The sum of the number of times TCP connections have made a direct transition to a CLOSED state from either the SYN-SENT state or the SYN-RCVD state, or a LISTEN state from the SYN-RCVD state.

**TCP connections currently established.** The number if TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT.

**TCP established and then reset.** The number of times TCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state.

**TCP passive opens.** The number of times TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state. This number is an indication of the number of times a remote system opened a connection to this system.

**TCP reset segments.** The number of TCP segments sent containing the RST flag.

**TCP retransmitted segments.** The number of TCP segments transmitted containing one or more previously transmitted octets.

**TCP segments received.** The total number of segments received, including those received in error. This count includes segments received on currently established connections.

**TCP segments received in error.** The total number of segments received in error (for example, bad TCP checksums).

**TCP segments sent.** The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets.

**UDP datagrams not delivered application port not found.** The total number of received UDP datagrams for UDP users for which there was no application at the destination port.

**UDP datagrams not delivered other datagrams in error.** The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port.

**UDP datagrams received.** The total number of segments received, including those received in error. This count includes segments received on currently established connections.

**UDP datagrams sent.** The total number of UDP datagrams sent from this entity.

## NCND0200 Format

This format returns detailed information about the TCP connection status in addition to the TCP/IPv4 connection totals (format NCND0100). For detailed descriptions of the fields in the table, see “Field Descriptions” on page 183.

Offset		Type	Field
Dec	Hex		
0	0		Returns everything from format NCND0100
Decimal and hexadecimal offsets are reached by using the offset to additional information field in format NCND0100. This applies to all entries below.		BINARY(4)	Protocol
		BINARY(4)	Local IP address
		BINARY(4)	Local port number
		BINARY(4)	Remote IP address
		BINARY(4)	Remote port number
		BINARY(4)	Round-trip time
		BINARY(4)	Round-trip variance
		BINARY(4)	Outgoing bytes buffered
		BINARY(4)	User send next
		BINARY(4)	Send next
		BINARY(4)	Send unacknowledged
		BINARY(4)	Outgoing push number
		BINARY(4)	Outgoing urgency number
		BINARY(4)	Outgoing window number
		BINARY(4)	Incoming bytes buffered
		BINARY(4)	Receive next
		BINARY(4)	User receive next
		BINARY(4)	Incoming push number
		BINARY(4)	Incoming urgency number
		BINARY(4)	Incoming window number
		BINARY(4)	Total retransmissions
		BINARY(4)	Current retransmissions
		BINARY(4)	Maximum window size
		BINARY(4)	Current window size
		BINARY(4)	Last update
		BINARY(4)	Last update acknowledged
		BINARY(4)	Congestion window

Offset		Type	Field
Dec	Hex		
		BINARY(4)	Slow start threshold
		BINARY(4)	Maximum segment size
		BINARY(4)	Initial send sequence number
		BINARY(4)	Initial receive sequence number
		BINARY(4)	Connection transport layer
		BINARY(4)	TCP state
		BINARY(4)	Connection open type
		BINARY(4)	Idle time in milliseconds
		CHAR(40)	IP options
		BINARY(4)	Bytes in
		BINARY(4)	Bytes out
		BINARY(4)	Socket state
		BINARY(4)	Offset to list of socket options associated with connection
		BINARY(4)	Number of socket options associated with connection
		BINARY(4)	Entry length for list of socket options associated with connection
		BINARY(4)	Offset to list of jobs associated with connection
		BINARY(4)	Number of jobs associated with connection
		BINARY(4)	Entry length for list of jobs associated with connection
		CHAR(10)	Associated user profile
		CHAR(2)	Reserved

## List of Socket Options.

These fields repeat for each socket option.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Socket option
4	4	BINARY(4)	Option value

## List of Jobs/Tasks Associated with this Connection.

These fields repeat for each job or task.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Format entry
4	4	CHAR(16)	Task name
20	14	CHAR(10)	Job name
30	1E	CHAR(10)	Job user name
40	28	CHAR(6)	Job number
46	2E	CHAR(16)	Internal job identifier

## Field Descriptions

**Associated user profile.** The user profile of the job on the local system which first performed a sockets API bind() of the socket.

**Note:** This field does not reliably indicate the current user of a connection or socket. To see a list of the jobs or tasks currently using a connection or socket, use the List of Jobs/Tasks Associated with this Connection.

**Bytes in.** The total number of bytes received on the connection, including framing characters.

**Bytes out.** The total number of bytes transmitted on the connection, including framing characters.

**Congestion window.** The number of segments that are sent on the next transmission. If an acknowledgment is received, the number is increased. If an acknowledgment is not received, the number is reset to the smallest allowable number. This field is only valid for TCP connections.

**Connection open type.** A TCP connection can be opened in the following ways:

- |   |   |
|---|---|
| 0 | Passive - A remote host opens the connection.                 |
| 1 | Active - The local system opens the connection.               |
| 2 | Unsupported - Connection open type not supported by protocol. |

**Connection transport layer.** The transport that a connection is using:

- |   |        |
|---|--------|
| 0 | IPS    |
| 1 | IPX    |
| 2 | TCP/IP |
- Note:** As of V5R2, IPX is no longer supported.

**Current retransmissions.** The number of times the local system retransmitted the current segment without receiving an acknowledgment. This is sometimes referred to as the 'backoff count. This field is only valid for TCP connections.

**Current window size.** The current send window size in bytes. This field is only valid for TCP connections.

**Entry length for list of jobs associated with connection.** The entry length in bytes of each element in the list of job connections returned with this format. A value of zero is returned if the list is empty.

**Entry length for list of socket options associated with connection.** The entry length in bytes of each element in the list of socket options returned with this format. A value of zero is returned if the list is empty.

**Format entry.** Type of list format for job or task connections.

- |   |   |
|---|---|
| 1 | Represents a job format. For this format the task name will be blank.   |
| 2 | Represents a task format. For this format the job name, username, number and internal identifier will be blank. |

**Idle time.** The length of time since the last activity on this connection. The length of time is returned in milliseconds.

**Incoming bytes buffered.** The current number of bytes that are received and buffered by TCP. These bytes are available to be read by an application.

**Incoming push number.** The sequence number of the last byte of pushed data in the incoming data stream. This value is zero if no push data is in the incoming data stream. This field is only valid for TCP connections.

**Incoming urgency number.** The sequence number of the last byte of urgent data in the incoming data stream. This value is zero if no urgent data is in the incoming data stream. This field is only valid for TCP connections.

**Incoming window number.** The largest sequence number in the incoming window of this connection. Data bytes in the incoming stream having sequence numbers larger than this number are not accepted. This field is only valid for TCP connections.

**Initial receive sequence number.** The first sequence number received on this connection. This field is only valid for TCP connections.

**Initial send sequence number.** The first sequence number sent on this connection. This field is only valid for TCP connections.

**IP options.** Used in displaying the IP datagram options that may have been specified for a connection.

**Internal job identifier.** A value sent to other APIs to speed the process of locating the job on the system.

**Job name.** The simple job name as identified to the system.

**Job number.** System-assigned job or task number.

**Job user name.** The user name identifies the user profile under which the job is started. The following special value may be returned:

\*SIGNON

This connection is a telnet connection and the system is performing sign-on processing or is displaying a sign-on prompt on it. In this case the Job name field will contain the network device name, the Job number and Internal job identifier fields will be empty.

**Last update.** The sequence number of the incoming segment used for the last window update that occurred on the connection. This field is only valid for TCP connections.

**Last update acknowledged.** The acknowledgment number of the incoming segment used for the last window update that occurred on the connection. This field is only valid for TCP connections.

**Local IP address.** The local address of this connection on this system.

**Local port number.** Your local system port number.

**Maximum segment size.** The size in bytes of the largest segment that may be transmitted on this connection. This field is only valid for TCP connections.

**Maximum window size.** The largest size of the send window, in bytes, during the entire time the connection has been active. This field is only valid for TCP connections.

**Number of jobs associated with connection.** The number of elements in the list of job connections returned with this format. A value of zero is returned if the list is empty.



**Number of socket options associated with connection.** The number of elements in the list of socket options returned with this format. A value of zero is returned if the list is empty.

**Offset to list of jobs associated with connection.** The offset in bytes to the first element in the list of job connections returned with this format. A value of zero is returned if the list is empty.

**Offset to list of socket options associated with connection.** The offset in bytes to the first element in the list of socket options returned with this format. A value of zero is returned if the list is empty.

**Option value.** The value returned for a particular socket option. Option is set if a nonzero value is returned.

**Outgoing bytes buffered.** The current number of bytes that an application has requested to send, but TCP has not yet sent. If TCP has sent the bytes to the remote system but has not yet received an acknowledgment, the bytes are considered 'not sent'. They are included in this count.

**Outgoing push number.** The sequence number of the last byte of push data in the outgoing stream. This value is zero if no push data is in the outgoing data stream. This field is only valid for TCP connections.

**Outgoing urgency number.** The sequence number of the last byte of urgent data in the outgoing data stream. This value is zero if no urgent data is in the outgoing data stream. This field is only valid for TCP connections.

**Outgoing window number.** The largest sequence number in the send window of the connection. The local TCP application cannot send data bytes with sequence numbers greater than the outgoing window number.

**Protocol.** Identifies the type of connection protocol.

- |   |   |
|---|---|
| 1 | TCP - A Transmission Control Protocol (TCP) connection or socket. |
| 2 | UDP - A User Datagram Protocol (UDP) socket.                      |

**Receive next.** The next sequence number the local TCP is expecting to receive.

**Remote IP address.** The internet address of the remote host. Zero is shown, if the list entry is for a UDP socket.

**Remote port number.** The remote host port number. Zero is shown, if the list entry is for a UDP socket.

**Round-trip time.** The smoothed round-trip time interval in milliseconds. This is a measure of the time required for a segment on the connection to arrive at its destination, to be processed, and to return an acknowledgment to the client. This field is only valid for TCP connections.

**Round-trip variance.** The variance in milliseconds from the previous round-trip time. This field is only valid for TCP connections.

**Send next.** The sequence number of the next byte of data that the local TCP application sends to the remote TCP application.

**Send unacknowledged.** The sequence number of the last segment sent that was not acknowledged. This is the smallest sequence number of the send window. This field is only valid for TCP connections.

**Slow start threshold.** The current values for the slow-start threshold and the congestion window are indirect indicators of the flow of data through a TCP connection. These values are used by TCP as part of a congestion control algorithm. This algorithm ensures that this system sends data at a slow rate at first.

After the first data has been successfully sent, the rate in which data is sent increases. This change is made in a controlled manner that is dependent on the amount of congestion in the network. Congestion control occurs both at connection start time and when congestion is detected. The values used for the slow-start threshold and the congestion window are determined by TCP and cannot be set by the user.

**Socket option.** Socket options for this connection.

1	<b>Socket broadcast option</b> Determine if messages can be sent to the broadcast address. This option is only supported for sockets with an address family of AF_INET and type SOCK_DGRAM or SOCK_RAW. Option is set if a nonzero value is returned.
2	<b>Socket bypass route option</b> - Determine if the normal routing mechanism is being bypassed. This option is only supported by sockets with an address family of AF_INET or AF_INET6. Option is set if a nonzero value is returned.
3	<b>Socket debug option</b> - Determine if low-level debugging is active. Option is set if a nonzero value is returned.
4	<b>Socket error</b> - Return any pending errors in the socket. The value returned corresponds to the standard error codes.
5	<b>Socket keep alive option</b> - Determine if the connection is being kept up by periodic transmissions. This option is only supported for sockets with an address family of AF_INET or AF_INET6 and type SOCK_STREAM. Option is set if a nonzero value is returned.
6	<b>Socket linger option</b> - Determine whether the system attempts to deliver any buffered data or if the system discards it when a close() is issued. For sockets that are using a connection-oriented transport service with an address family of AF_INET or AF_INET6, the default is off (which means that the system attempts to send any queued data, with an infinite wait-time).
7	<b>Socket linger time</b> - Determine how much time in seconds the system will wait to send buffered data.
8	<b>Socket out-of-band data option</b> - Determine if out-of-band data is received inline with normal data. This option is only supported for sockets with an address family of AF_INET or AF_INET6. Option is set if a nonzero value is returned.
9	<b>Socket receive buffer size</b> - Determine the size of the receive buffer.
10	<b>Socket receive low-water mark size</b> - Determine the size of the receive low-water mark. The default size is 1. This option is only supported for sockets with type SOCK_STREAM.
11	<b>Socket reuse address option</b> - Determine if the local socket address can be reused. This option is only supported by sockets with an address family of AF_INET or AF_INET6 and with type SOCK_STREAM or SOCK_DGRAM. Option is set if a nonzero value is returned.
12	<b>Socket send buffer size</b> - Determine the size of the send buffer.
13	<b>Socket type value</b> - Determine the value for the socket type.
	1 Stream type.
	2 Datagram type.
	3 Raw type.
	4 Sequential packet type.

**Socket state.** The current state of the socket.

0	Uninitialized
1	Unbound
2	Bound
3	Listening
4	Connecting

5	Connected
6	Disconnected
7	Error

**Task name.** The task name as identified to the system.

**TCP state.** A typical connection goes through the states:

0	Listen, waiting for a connection request from any remote host.
1	SYN-sent, waiting for a matching connection request after having sent connection request.
2	SYN-received, waiting for a confirming connection request acknowledgement.
3	Established, the normal state in which data is transferred.
4	FIN-wait-1, waiting for the remote host to acknowledge the local system request to end the connection.
5	FIN-wait-2, waiting for the remote host request to end the connection.
6	Close-wait, waiting for an end connection request from the local user.
7	Closing, waiting for an end connection request acknowledgement from the remote host.
8	Last-ACK, waiting for the remote host to acknowledge an end connection request.
9	Time-wait, waiting to allow the remote host enough time to receive the local system's acknowledgement to end the connection.
10	Closed, the connection has ended.
11	State value not supported by protocol.

**Total retransmissions.** The total number of times the local system retransmitted a segment because an acknowledgment was not received. This is a cumulative count of all segments resent during the entire time the connection has been active. This field is only valid for TCP connections.

**User send next.** The sequence number of the next byte of data to be sent by the client application. This field is only valid for TCP connections.

**User receive next.** The sequence number of the next byte to be passed to the application by TCP.

## NCND1100 Format

Format NCND1100 returns information regarding the TCP/IP<sub>v6</sub> connection totals. For detailed descriptions of the fields in the table, see "Field Descriptions" on page 188.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	TCP connections currently established
12	C	BINARY(4)	TCP active opens
16	10	BINARY(4)	TCP passive opens
20	14	BINARY(4)	TCP attempted opens that failed
24	18	BINARY(4)	TCP established and then reset
28	1C	BINARY(4)	TCP segments sent
32	20	BINARY(4)	TCP retransmitted segments

Offset		Type	Field
Dec	Hex		
36	24	BINARY(4)	TCP reset segments
40	28	BINARY(4)	TCP segments received
44	2C	BINARY(4)	TCP segments received in error
48	30	BINARY(4)	UDP datagrams sent
52	34	BINARY(4)	UDP datagrams received
56	38	BINARY(4)	UDP datagrams not delivered - application port not found
60	3C	BINARY(4)	UDP datagrams not delivered - other datagrams in error
64	40	BINARY(4)	Offset to additional information
68	44	BINARY(4)	Length of additional information

## Field Descriptions

**Bytes available.** All of the available bytes for use in your application.

**Bytes returned.** The number of bytes returned to the user. This may be some but not all the bytes available.

**Length of Additional Information.** The length in bytes of additional information returned that is not part of format NCND0100.

**Offset to Additional Information.** The offset in bytes to the rest of the information if a format other than NCND0100 is requested.

**TCP active opens.** The number of times TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state. This number is an indication of the number of times this local system opened a connection to a remote system.

**TCP attempted opens that failed.** The sum of the number of times TCP connections have made a direct transition to a, CLOSED state from either the SYN-SENT state or the SYN-RCVD state, or a LISTEN state from the SYN-RCVD state.

**TCP connections currently established.** The number if TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT.

**TCP established and then reset.** The number of times TCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state.

**TCP passive opens.** The number of times TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state. This number is an indication of the number of times a remote system opened a connection to this system.

**TCP reset segments.** The number of TCP segments sent containing the RST flag.

**TCP retransmitted segments.** The number of TCP segments transmitted containing one or more previously transmitted octets.

**TCP segments received.** The total number of segments received, including those received in error. This count includes segments received on currently established connections.

**TCP segments received in error.** The total number of segments received in error (for example, bad TCP checksums).

**TCP segments sent.** The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets.

**UDP datagrams not delivered - application port not found.** The total number of received UDP datagrams for UDP users for which there was no application at the destination port.

**UDP datagrams not delivered - other datagrams in error.** The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port.

**UDP datagrams received.** The total number of segments received, including those received in error. This count includes segments received on currently established connections.

**UDP datagrams sent.** The total number of UDP datagrams sent from this entity.

## NCND1200 Format

This format returns detailed information about the TCP connection status in addition to the TCP/IPv6 connection totals (format NCND1100). For detailed descriptions of the fields in the table, see “Field Descriptions” on page 191.

Offset		Type	Field
Dec	Hex		
0	0		Returns everything from format NCND1100
Decimal and hexadecimal offsets are reached by using the offset to additional information field in format NCND1100. This applies to all entries below.		BINARY(4)	Protocol
		CHAR(16)	Local IPv6 address
		BINARY(4)	Local port number
		CHAR(16)	Remote IPv6 address
		BINARY(4)	Remote port number
		BINARY(4)	Round-trip time
		BINARY(4)	Round-trip variance
		BINARY(4)	Outgoing bytes buffered
		BINARY(4)	User send next
		BINARY(4)	Send next
		BINARY(4)	Send unacknowledged
		BINARY(4)	Outgoing push number
		BINARY(4)	Outgoing urgency number
		BINARY(4)	Outgoing window number
		BINARY(4)	Incoming bytes buffered
		BINARY(4)	Receive next
		BINARY(4)	User receive next
		BINARY(4)	Incoming push number
		BINARY(4)	Incoming urgency number
		BINARY(4)	Incoming window number
		BINARY(4)	Total retransmissions
		BINARY(4)	Current retransmissions
		BINARY(4)	Maximum window size
		BINARY(4)	Current window size

Offset		Type	Field
Dec	Hex		
		BINARY(4)	Last update
		BINARY(4)	Last update acknowledged
		BINARY(4)	Congestion window
		BINARY(4)	Slow start threshold
		BINARY(4)	Maximum segment size
		BINARY(4)	Initial send sequence number
		BINARY(4)	Initial receive sequence number
		BINARY(4)	Connection transport layer
		BINARY(4)	TCP state
		BINARY(4)	Connection open type
		BINARY(4)	Idle time
		BINARY(8)	Bytes in
		BINARY(8)	Bytes out
		BINARY(4)	Socket state
		CHAR(10)	Associated user profile
		CHAR(2)	Reserved
		BINARY(4)	Offset to list of socket options associated with connection
		BINARY(4)	Number of socket options associated with connection
		BINARY(4)	Entry length for list of socket options associated with connection
		BINARY(4)	Offset to list of jobs associated with connection
		BINARY(4)	Number of jobs associated with connection
		BINARY(4)	Entry length for list of jobs associated with connection

## List of Socket Options.

These fields repeat for each socket option.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Socket option
4	4	BINARY(4)	Option value

## List of Jobs/Tasks Associated with this Connection.

These fields repeat for each job or task.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Entry type
4	4	CHAR(16)	Task name
20	14	CHAR(10)	Job name
30	1E	CHAR(10)	Job user name
40	28	CHAR(6)	Job number
46	2E	CHAR(16)	Internal job identifier

## Field Descriptions

**Associated user profile.** The user profile of the job on the local system which first performed a sockets API bind() of the socket.

**Note:** This field does not reliably indicate the current user of a connection or socket. To see a list of the jobs or tasks currently using a connection or socket, use the List of Jobs/Tasks Associated with this Connection.

**Bytes in.** The total number of bytes received on the connection, including framing characters.

**Bytes out.** The total number of bytes transmitted on the connection, including framing characters.

**Congestion window.** The number of segments that are sent on the next transmission. If an acknowledgment is received, the number is increased. If an acknowledgment is not received, the number is reset to the smallest allowable number. This field is only valid for TCP connections.

**Connection open type.** The method in which the TCP connection was opened.

Possible values are:

- |   |   |
|---|---|
| 0 | Passive - A remote host opened the connection.                |
| 1 | Active - The local system opened the connection.              |
| 2 | Unsupported - Connection open type not supported by protocol. |

**Connection transport layer.** The transport that the connection is using.

Possible values are:

- |   |        |
|---|--------|
| 0 | IPS    |
| 1 | IPX    |
| 2 | TCP/IP |
- Note:** As of V5R2, IPX is no longer supported.

**Current retransmissions.** The number of times the local system retransmitted the current segment without receiving an acknowledgment. This is sometimes referred to as the 'backoff count'. This field is only valid for TCP connections.

**Current window size.** The current send window size in bytes. This field is only valid for TCP connections.

**Entry length for list of jobs associated with connection.** The entry length in bytes of each element in the list of job connections returned with this format. A value of zero is returned if the list is empty.

**Entry length for list of socket options associated with connection.** The entry length in bytes of each element in the list of socket options returned with this format. A value of zero is returned if the list is empty.

**Entry type.** Specifies whether this entry is a job or a task.

Possible values are:

- |   |   |
|---|---|
| 1 | Represents a job format. For this format the task name field is not applicable.   |
| 2 | Represents a task format. For this format the job name, username, number and internal job identifier fields are not applicable. |

**Idle time.** The length of time since the last activity on this connection. The length of time is returned in milliseconds.

**Incoming bytes buffered.** The current number of bytes that are received and buffered by TCP. These bytes are available to be read by an application.

**Incoming push number.** The sequence number of the last byte of pushed data in the incoming data stream. This value is zero if no push data is in the incoming data stream. This field is only valid for TCP connections.

**Incoming urgency number.** The sequence number of the last byte of urgent data in the incoming data stream. This value is zero if no urgent data is in the incoming data stream. This field is only valid for TCP connections.

**Incoming window number.** The largest sequence number in the incoming window of this connection. Data bytes in the incoming stream having sequence numbers larger than this number are not accepted. This field is only valid for TCP connections.

**Initial receive sequence number.** The first sequence number received on this connection. This field is only valid for TCP connections.

**Initial send sequence number.** The first sequence number sent on this connection. This field is only valid for TCP connections.

**Internal job identifier.** A value sent to other APIs to speed the process of locating the job on the system.

**Job name.** The simple job name as identified to the system.

**Job number.** System-assigned job or task number.

**Job user name.** The user name identifies the user profile under which the job is started.

The following special value may be returned:

\**SIGNON*            This connection is a telnet connection and the system is performing sign-on processing or is displaying a sign-on prompt on it. In this case the Job name field will contain the network device name, and the Job number and Internal job identifier fields will be empty.

**Last update.** The sequence number of the incoming segment used for the last window update that occurred on the connection. This field is only valid for TCP connections.

**Last update acknowledged.** The acknowledgment number of the incoming segment used for the last window update that occurred on the connection. This field is only valid for TCP connections.

**Local IPv6 address.** The local system internet address, in IPv6 address format, of the connection. Even though this field is defined as a character field, a binary IPv6 address is returned in it.

**Local port number.** The port number of the local end of the connection.

**Maximum segment size.** The size in bytes of the largest segment that may be transmitted on this connection. This field is only valid for TCP connections.

**Maximum window size.** The largest size of the send window, in bytes, during the entire time the connection has been active. This field is only valid for TCP connections.



**Number of jobs associated with connection.** The number of elements in the list of job connections returned with this format. A value of zero is returned if the list is empty.

**Number of socket options associated with connection.** The number of elements in the list of socket options returned with this format. A value of zero is returned if the list is empty.

**Offset to list of jobs associated with connection.** The offset in bytes to the first element in the list of job connections returned with this format. A value of zero is returned if the list is empty.

**Offset to list of socket options associated with connection.** The offset in bytes to the first element in the list of socket options returned with this format. A value of zero is returned if the list is empty.

**Option value.** The value returned for a particular socket option. The socket option is set if a nonzero value is returned.

**Outgoing bytes buffered.** The current number of bytes that an application has requested to send, but TCP has not yet sent. If TCP has sent the bytes to the remote system but has not yet received an acknowledgment, the bytes are considered 'not sent'. They are included in this count.

**Outgoing push number.** The sequence number of the last byte of push data in the outgoing stream. This value is zero if no push data is in the outgoing data stream. This field is only valid for TCP connections.

**Outgoing urgency number.** The sequence number of the last byte of urgent data in the outgoing data stream. This value is zero if no urgent data is in the outgoing data stream. This field is only valid for TCP connections.

**Outgoing window number.** The largest sequence number in the send window of the connection. The local TCP application cannot send data bytes with sequence numbers greater than the outgoing window number.

**Protocol.** Identifies the type of connection protocol.

- |   |  |
|---|--|
| 1 | <b>TCP</b> - A Transmission Control Protocol (TCP) connection or socket. |
| 2 | <b>UDP</b> - A User Datagram Protocol (UDP) socket.                      |

**Receive next.** The next sequence number that TCP is expecting to receive.

**Remote IPv6 address.** The local system internet address, in IPv6 address format, of the connection. Even though this field is defined as a character field, a binary IPv6 address is returned in it.

The following special value may be returned:

- |   |   |
|---|---|
| 0 | This "connection" is a listening socket, and there is no remote IPv6 address. The zero is returned as a series of binary NULLs (x'000000...') |
|---|---|

**Remote port number.** The port number of the remote end of the connection.

The following special value may be returned:

- |   |   |
|---|---|
| 0 | This "connection" is a listening socket and there is no remote port number. |
|---|---|

**Reserved.** An ignored field.

**Round-trip time.** The smoothed round-trip time interval in milliseconds. This is a measure of the time required for a segment on the connection to arrive at its destination, to be processed, and to return an acknowledgment to the client. This field is only valid for TCP connections.

**Round-trip variance.** The variance in milliseconds from the previous round-trip time. This field is only valid for TCP connections.

**Send next.** The sequence number of the next byte of data that the local TCP application sends to the remote TCP application.

**Send unacknowledged.** The sequence number of the last segment sent that was not acknowledged. This is the smallest sequence number of the send window. This field is only valid for TCP connections.

**Slow start threshold.** The current values for the slow-start threshold and the congestion window are indirect indicators of the flow of data through a TCP connection. These values are used by TCP as part of a congestion control algorithm. This algorithm ensures that this system sends data at a slow rate at first. After the first data has been successfully sent, the rate in which data is sent increases. This change is made in a controlled manner that is dependent on the amount of congestion in the network. Congestion control occurs both at connection start time and when congestion is detected. The values used for the slow-start threshold and the congestion window are determined by TCP and cannot be set by the user.

**Socket option.** Socket options for this connection.

- |    |  |
|----|--|
| 1  | <b>Socket broadcast option</b> Determine if messages can be sent to the broadcast address. This option is only supported for sockets with an address family of AF_INET and type SOCK_DGRAM or SOCK_RAW. Option is set if a nonzero value is returned.  |
| 2  | <b>Socket bypass route option</b> - Determine if the normal routing mechanism is being bypassed. This option is only supported by sockets with an address family of AF_INET or AF_INET6. Option is set if a nonzero value is returned.   |
| 3  | <b>Socket debug option</b> - Determine if low-level debugging is active. Option is set if a nonzero value is returned.   |
| 4  | <b>Socket error</b> - Return any pending errors in the socket. The value returned corresponds to the standard error codes.   |
| 5  | <b>Socket keep alive option</b> - Determine if the connection is being kept up by periodic transmissions. This option is only supported for sockets with an address family of AF_INET or AF_INET6 and type SOCK_STREAM. Option is set if a nonzero value is returned.  |
| 6  | <b>Socket linger option</b> - Determine whether the system attempts to deliver any buffered data or if the system discards it when a close() is issued. For sockets that are using a connection-oriented transport service with an address family of AF_INET or AF_INET6, the default is off (which means that the system attempts to send any queued data, with an infinite wait-time). |
| 7  | <b>Socket linger time</b> - Determine how much time in seconds the system will wait to send buffered data.   |
| 8  | <b>Socket out-of-band data option</b> - Determine if out-of-band data is received inline with normal data. This option is only supported for sockets with an address family of AF_INET or AF_INET6. Option is set if a nonzero value is returned.  |
| 9  | <b>Socket receive buffer size</b> - Determine the size of the receive buffer.  |
| 10 | <b>Socket receive low-water mark size</b> - Determine the size of the receive low-water mark. The default size is 1. This option is only supported for sockets with type SOCK_STREAM.  |
| 11 | <b>Socket reuse address option</b> - Determine if the local socket address can be reused. This option is only supported by sockets with an address family of AF_INET or AF_INET6 and with type SOCK_STREAM or SOCK_DGRAM. Option is set if a nonzero value is returned.  |

12  
13

**Socket send buffer size** - Determine the size of the send buffer.  
**Socket type value** - Determine the value for the socket type.

- 1 Stream type.
- 2 Datagram type.
- 3 Raw type.
- 4 Sequential packet type.

**Socket state.** The current state of the socket.

Possible values are:

- 0 Uninitialized
- 1 Unbound
- 2 Bound
- 3 Listening
- 4 Connecting
- 5 Connected
- 6 Disconnected
- 7 Error

**Task name.** The task name as identified to the system.

**TCP state.** A typical connection goes through the states:

- 0 Listen, waiting for a connection request from any remote host.
- 1 SYN-sent, waiting for a matching connection request after having sent connection request.
- 2 SYN-received, waiting for a confirming connection request acknowledgement.
- 3 Established, the normal state in which data is transferred.
- 4 FIN-wait-1, waiting for the remote host to acknowledge the local system request to end the connection.
- 5 FIN-wait-2, waiting for the remote host request to end the connection.
- 6 Close-wait, waiting for an end connection request from the local user.
- 7 Closing, waiting for an end connection request acknowledgement from the remote host.
- 8 Last-ACK, waiting for the remote host to acknowledge an end connection request.
- 9 Time-wait, waiting to allow the remote host enough time to receive the local system's acknowledgement to end the connection.
- 10 Closed, the connection has ended.
- 11 State value not supported by protocol.

**Total retransmissions.** The total number of times the local system retransmitted a segment because an acknowledgment was not received. This is a cumulative count of all segments resent during the entire time the connection has been active. This field is only valid for TCP connections.

**User send next.** The sequence number of the next byte of data to be sent by the client application. This field is only valid for TCP connections.

**User receive next.** The sequence number of the next byte to be passed to the application by TCP.

## Error Messages

Message ID	Error Message Text
TCP84C0 E	TCP/IP stack not active.
TCP84C5 E	Error providing TCP/IP Network Status list information.
TCP84C6 E	Internal operations error - RESULT &1 CC &2 RC &3 ERRNO &4.
TCP84C9 I	Information returned incomplete.

Message ID	Error Message Text
TCP84CA E	Connection request parameter not valid.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C1E E	Required parameter &1 omitted.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF3CF2 E	API contains a problem. See prior messages to determine why the failure occurred.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R1

[Top](#) | ["Communications APIs," on page 1](#) | [APIs by category](#)

---

## Retrieve PPP Connection Profiles (QtocRtvPPPCnnPrf) API

Required Parameter Group:	
<b>1</b>	Receiver variable
<b>Output</b>	Char(*)
<b>2</b>	Length of receiver variable
<b>Input</b>	Binary(4)
<b>3</b>	Profile name
<b>Input</b>	Char(10)
<b>4</b>	Format name
<b>Input</b>	Char(8)
<b>5</b>	Error Code
<b>I/O</b>	Char(*)
Service Program: QTOCPPPAPI	
Threadsafe: Yes	

The Retrieve PPP Connection Profiles (QtocRtvPPPCnnPrf) API retrieves the details of a specific PPP connection job profile. If the connection profile describes multiple connections, then details of each connection are also retrieved.

### Authorities and Locks

None.

### Required Parameter Group

#### Receiver variable

OUTPUT; CHAR(\*)

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

**Length of receiver variable**

INPUT; BINARY(4)

The length of the receiver variable. If the length is larger than the size of the receiver variable, the results may not be predictable. The minimum length is 8 bytes.

**Profile name**

INPUT; CHAR(10)

The name of the PPP connection profile to be returned.

**Format name**

INPUT; CHAR(8)

The format of the retrieved profile to be returned. The format names supported are:

*PRFR0100*

Connection profile attributes. Refer to “PRFR0100 Format” for details on the format.

*PRFR0200*

Connection profile static parameters. Refer to “PRFR0100 Format” and “PRFR0200 Format” on page 199 for details on the format.

**Error code**

I/O; CHAR(\*)

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

**Format of Connection Profile Attributes Information**

To retrieve the basic connection profile information and current profile job status, use format PRFR0100. For more detailed profile and connection attributes, use format PRFR0200.

**PRFR0100 Format**

The following data about a connection profile is returned for the PRFR0100 format. For detailed descriptions of the fields in the table, see “Field Descriptions” on page 198.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Profile mode
12	C	BINARY(4)	Connection protocol
16	10	BINARY(4)	Connection status
20	14	BINARY(4)	Connection type
24	18	BINARY(4)	Profile job type
28	1C	BINARY(4)	➤ Automatic start ⬅
32	20	CHAR(10)	Profile name
42	2A	CHAR(50)	Profile description
92	5C	CHAR(16)	Reserved
108	6C	BINARY(4)	Offset to additional information
112	70	BINARY(4)	Length of additional information

## Field Descriptions



**Automatic start.** Whether the profile is started automatically when the TCP/IP stack is activated. Possible values are:

- 0 NO. This profile is not started automatically.
- 1 YES. This profile is started automatically.



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

**Bytes returned.** The number of bytes of data returned.

**Connection protocol.** The type of point-to-point connection provided by the profile job.

- 1 SLIP.
- 2 PPP.

**Connection status.** The current connection of job status of the profile job. Values are as follows:

- 1 Inactive
- 2 Session error
- 3 Ended - information available
- 4 Session start submitted
- 11 Session job starting
- 12 Session job ending
- 13 Session ended - job log pending
- 14 Adding TCP/IP configuration
- 15 Removing TCP/IP configuration
- 16 Message pending
- 17 Session error
- 18 Starting TCP/IP
- 19 Ending TCP/IP
- 21 Calling remote system
- 22 Waiting for incoming call
- 23 Connecting
- 24 Active
- 26 Switched line-dial on demand
- 27 Waiting for incoming call - switched line-answer enabled dial on demand
- 28 Waiting for shared line resource
- 29 Requesting shared line resource
- 31 LCP initializing
- 32 LCP starting
- 33 LCP closing
- 34 LCP closed
- 35 LCP waiting for configuration request
- 36 LCP configuring
- 37 LCP authenticating
- 41 IPCP initializing
- 42 IPCP starting
- 43 IPCP ending
- 44 IPCP stopped
- 45 IPCP waiting for configuration request
- 46 IPCP configuring

47	IPCP opening
51	Multi-connection - waiting for incoming call(s)
52	Multi-connection L2TP initiator waiting for tunnel
53	Multi-connection - at least one connection active
54	Multi-hop terminator starting multi-hop initiator
55	Multi-hop initiator establishing second hop tunnel
56	Multi-hop initiator tunnel pre-started
57	Multi-hop connection active
58	Starting VPN connection
59	Negotiating IPSEC SA
60	PPPoE discovery stage
61	PPPoE session stage

**Connection type.** The type of connection provided by the profile job. Values are:

1	Switched or dialed connection
2	Leased or non-switched connection
3	Virtual circuit connection

**Length of additional information.** The length in bytes of additional information returned that is not part of format PRFR0100.

**Offset to additional information.** The offset in bytes to the rest of the information if a format other than PRFR0100 is requested.

**Profile description.** The text description of the function performed by this profile connection job..

**Profile job type.** The type of job support required for the profile.

1	Single connection profile
2	Multi-connection or multilink connection profile

**Profile mode.** The function provided by the profile job. Values are:

1	Dial only.
2	Answer only.
3	Dial-on-demand.
4	Answer enabled dial-on-demand.
5	L2TP virtual initiator.
6	Remote peer enabled dial-on-demand.
7	L2TP initiator-on-demand.
8	L2TP multihop initiator.
9	PPPoE initiator.

**Profile name.** The name of this connection profile description.

## PRFR0200 Format

The following data about a connection profile is returned for the PRFR0200 format. For detailed descriptions of the fields in the table, see “Field Descriptions” on page 201.

Offset		Type	Field
Dec	Hex		
0	0		Returns everything from format PRFR0100

Offset		Type	Field
Dec	Hex		
Decimal and hexadecimal offsets are reached by using the offset to additional information field in format PRFR0100. This applies to all entries below.		BINARY(4)	Move current remote phone number if dial operation is successful
		BINARY(4)	Redial when disconnected
		BINARY(4)	Number of dial attempts
		BINARY(4)	Delay between dial attempts
		BINARY(4)	Maximum number of connections
		BINARY(4)	Multilink connection enabled
		BINARY(4)	Maximum number of multilink connections
		BINARY(4)	Inactivity timeout
		BINARY(4)	Line definition
		CHAR(10)	Line name
		CHAR(10)	Line type
		CHAR(15)	L2TP tunnel end-point IP address
		CHAR(5)	Reserved
		BINARY(4)	Local user ID defined
		BINARY(4)	Local user ID encryption type
		CHAR(10)	Local user ID validation list name
		CHAR(6)	Reserved
		BINARY(4)	Remote user ID required for logon
		BINARY(4)	Remote user ID authentication protocols allowed
		BINARY(4)	Remote user ID validation method
		BINARY(4)	Use Radius for connection auditing and accounting
		CHAR(10)	Remote user ID validation list name
		CHAR(6)	Reserved
		BINARY(4)	ASCII CCSID of line data
		BINARY(4)	Connection script file defined
		CHAR(10)	Connection script library
		CHAR(10)	Connection script file
		CHAR(10)	Connection script member
		CHAR(2)	Reserved
		BINARY(4)	DNS definition
		CHAR(15)	DNS IP address
		CHAR(5)	Reserved
		BINARY(4)	Local IP address definition
		CHAR(15)	Local IP address
		CHAR(5)	Reserved
		BINARY(4)	Remote IP address definition
		CHAR(15)	Remote IP address (or start of range)
		CHAR(5)	Reserved
		BINARY(4)	Allow additional remote IP addresses by user ID
		BINARY(4)	Allow remote system to assign the remote IP address
		BINARY(4)	Allow IP datagram forwarding



Offset		Type	Field
Dec	Hex		
		BINARY(4)	Request VJ header compression
		BINARY(4)	Routing definition
		BINARY(4)	Hide address (full masquerading)
		BINARY(4)	Number of remote IP addresses
		CHAR(4)	Reserved
		CHAR(64)	Line pool list name
		CHAR(10)	Subsystem description
		CHAR(6)	Reserved
		BINARY(4)	Requires IP security protection
		CHAR(40)	IP security connection group
		CHAR(10)	Answer profile this dial-on-demand profile depends on
		CHAR(6)	Reserved
		BINARY(4)	Allow remote system to initiate call
		BINARY(4)	Allow BACP
		BINARY(4)	Add link percentage
		BINARY(4)	Time to wait (in seconds) for adding a link
		BINARY(4)	Drop link percentage
		BINARY(4)	Time to wait (in seconds) for dropping a link
		BINARY(4)	Bandwidth test direction
		BINARY(4)	Use filter rule
		CHAR(32)	Filter rule name
		BINARY(4)	Allow L2TP Multihop connections
		BINARY(4)	Allow L2TP outgoing call connections
		BINARY(4)	L2TP outgoing call line definition
		CHAR(10)	L2TP outgoing call line name
		CHAR(10)	Reserved
		BINARY(4)	Offset to profile detailed connection parameter entries
		BINARY(4)	Number of profile detailed connection parameter entries
		BINARY(4)	Entry length of profile detailed connection parameters
		BINARY(4)	Offset to remote phone number entries
		BINARY(4)	Number of remote phone number entries
		BINARY(4)	Entry length of remote phone numbers
		BINARY(4)	PPPoE server addressing
		BINARY(4)	Persistent PPPoE connection
		CHAR(256)	Requested PPPoE server name
		CHAR(256)	Requested PPPoE service

## Field Descriptions

**Add link percentage.** The percentage utilization of the connection before adding another link to a connection. Valid values are:

- 1
- 5
- 10
- 25

50  
75  
90 (default)  
95  
100

**Allow additional remote IP addresses by user ID.** Whether additional remote IP addresses may be specified for specific user ID entries. Valid values are:

0	No
1	Yes

**Allow BACP (Bandwidth Allocation Control Protocol).** Whether BACP is allowed/required for this connection. Valid values are:

0	No
1	Yes

**Allow IP datagram forwarding.** Whether IP datagrams not destined for this system should be forwarded. Valid values are:

0	No
1	Yes

**Allow L2TP Multihop connections.** Whether L2TP multihop connections are allowed by this profile connection job. Valid values are:

0	No
1	Yes

**Allow L2TP outgoing call connections.** Whether L2TP outgoing call connections are allowed by this profile connection job. Valid values are:

0	No
1	Yes

**Allow remote system to assign the remote IP address.** Whether the remote system is allowed to specify the remote IP address for the connection. Valid values are:

0	No
1	Yes

**Allow remote system to initiate call.** The remote system is allowed to initiate a call for an additional link for the connection. Valid values are:

0	No
1	Yes

**Answer profile this dial-on-demand profile depends on.** The name of the answer profile (connection job) that must be running to answer incoming connections before this profile connection job may be started.

**ASCII CCSID of line data.** The ASCII Coded Character Set ID of the line data for the connection that will be used to translate connection dialog to and from the EBCDIC character set of the Connection Script used by this profile connection job.

**Bandwidth test direction.** The data direction on the connection to test the bandwidth for adding and removing links. Valid values are:

0	N/A
1	Inbound and outbound
2	Outbound only

**Connection script file.** The name of the connection script file that is used by this profile connection job.

**Connection script file defined.** The connection script file that describes dialog for establishing a connection with the remote system. Valid values are:

0	No
1	Yes

**Connection script library.** The library containing a Connection Script file that is used by this profile connection job.

**Connection script member.** The member name of the Connection Script file that is used by this profile connection job.

**Delay between dial attempts.** The time (in seconds) to wait before next attempting to make a successful dialed connection. Valid values are:

1 - 60 (default = 15)

**DNS definition.** Whether a Domain Name Server IP address is to be added to the DNS address list when a connection is established for this profile connection job. Valid values are:

0	DNS not used - no address will be added
1	By IP address - the IP address is statically specified
2	Dynamic - the IP address will be supplied by the remote system

**DNS IP address.** The IP address of the Domain Name Server used by this profile connection job.

**Drop link percentage.** The percentage utilization of the connection before dropping a link of a connection. Valid values are:

1  
5  
10  
25  
40 (default)  
75  
90  
95  
100

**Entry length of profile detailed connection parameters.** The length in bytes of each profile detailed connection parameter entry returned for this profile. A value of zero is returned if the list is empty.

**Entry length of remote phone numbers.** The length in bytes of each remote phone number entry returned for this profile. A value of zero is returned if the list is empty.

**Filter rule name.** The name of the filter rule to be used by this connection profile.

**Hide address (full masquerading).** Whether all other IP addresses should be hidden by the IP address of the PPP connections established by this profile connection job. Valid values are:

0	No
1	Yes

**Inactivity timeout.** The value used for the inactivity timeout in the line description. Valid values are 15 - 65535 seconds.

**IP security connection group.** The name of the connection group that describes the IP Security details for connections established by this profile connection job.

**L2TP outgoing call line definition.** The line type to be used by this profile connection job for L2TP outgoing calls. Valid values are:

1	Single line
2	LinePool (single line)
5	ISDN line

**L2TP outgoing call line name.** The name of the line to be used by this profile connection job for L2TP outgoing calls.

**L2TP tunnel end-point address.** The IP address of the remote end of the tunnel for an L2TP initiator profile or the IP address of the local end of the tunnel for an L2TP terminator profile.

**Line definition.** The line selection method used by this profile connection job. Valid values are:

1	Specified line name
2	LinePool (single line)
3	LinePool (all)
5	ISDN line
6	L2TP line
7	PPPOE virtual line

**Line name.** Each TCP/IP interface is associated with a physical network. This field displays the name of the communications line description that identifies the physical network associated with an interface. May be blank when Line type selection is \*LINEPOOL and no member line has been selected.

**Line pool list name.** The name of the Line Pool list that contains the names of line descriptions available for use by this profile connection job.

**Line type.** The type of line connection defined in this connection profile. Possible values are:

*PPP	PPP line description
*POOL	Line name is a member of a line pool
*ISDN	ISDN line description
*L2TP	L2TP line description
*PPPOE	PPPoE line description
*ERROR	The selected line type is undefined or is improperly defined

**Local IP address.** The local IP address defined for connections established by this profile connection job.

**Local IP address definition.** How a local IP address is defined for connections established by this profile connection job. Valid values are:

- |   |  |
|---|--|
| 1 | By IP address - the IP address is statically specified             |
| 2 | Dynamic - the IP address will be negotiated with the remote system |

**Local user ID defined.** The User ID that is defined if authentication is required by the remote system. Valid values are:

- |   |     |
|---|-----|
| 0 | No  |
| 1 | Yes |

**Local user ID encryption type.** The encryption method for the local system user name and password when authenticating with the remote system. Valid values are:

- |   |           |
|---|-----------|
| 0 | Undefined |
| 1 | PAP only  |
| 2 | CHAP only |
| 3 | EAP only  |

**Local user ID validation list name.** The name of the validation list containing the local User ID and password when authenticating with the remote system.

**Maximum number of connections.** The maximum number of connections supported by this PPP job profile.

**Maximum number of multilink connections.** The maximum number of physical connections connections that can be bundled into a single multi-linked connection.

**Move current remote phone number if dial operation is successful.** Whether the current remote phone number should be moved if the call attempt is successful. Valid values are:

- |   |  |
|---|--|
| 0 | N/A  |
| 1 | Do NOT move number (default for non-multilink connections)             |
| 2 | Move number to the top of the list (default for multilink connections) |
| 3 | Move number to the bottom of the list                                  |

**Multilink connection enabled.** Whether multilink connections are enabled for the profile. Values are:

- |   |     |
|---|-----|
| 0 | No  |
| 1 | Yes |

**Number of profile detailed connection parameter entries.** The number of profile detailed connection parameter entries returned for this profile. A value of zero is returned if the list is empty.

**Number of remote phone number entries.** The number of remote phone number entries returned for this profile. A value of zero is returned if the list is empty.

**Number of remote IP addresses.** The number of IP addresses derived from the Remote IP start address defined for this profile connection job.

**Number of dial attempts.** The total number of dial attempts to achieve a successful connection.

**Offset to profile detailed connection parameter entries.** The offset from the beginning of the receiver variable, in bytes, to the first element in the profile detailed connection parameter entries returned for this profile. A value of zero is returned if the list is empty.

**Offset to remote phone number entries.** The offset from the beginning of the receiver variable, in bytes, to the first element in the remote phone number entries returned for this profile. A value of zero is returned if the list is empty.

**Persistent PPPoE connection.** Whether PPPoE connections for this profile are re-established when lost unexpectedly. Values are:

0	No
1	Yes

**PPPoE server addressing.** Describes the method used to select a PPPoE server connection. Valid values are:

0	Undefined
1	Connect to the default service of the first server that replies (default)
2	Connect to the default service of the requested server
3	Connect to the first server offering the requested service
4	Connect to the requested server offering the requested service

**Re-dial when disconnected.** Whether a dialed connection established by this profile connection job will be redialed if the connection is lost unexpectedly. Valid values are:

0	No
1	Yes

**Remote IP address definition.** How a remote IP address is defined for connections established by this profile connection job. Valid values are:

1	By IP address - the IP address is statically specified
2	Dynamic - the IP address will be negotiated with the remote system
3	Route specified - the IP address is specified by remote user
4	Address pool - the IP address will be selected from the address pool
5	DHCP - the IP address will be supplied by the DHCP server
6	Radius - the IP address will be supplied by the Radius server

**Remote IP address (or start of IP address pool).** The remote IP address (or starting IP address for multi-connection profiles) defined for connections established by this profile connection job.

**Remote user ID authentication protocols allowed.** The allowable protocols for remote user ID authentication. Valid values are:

0	N/A
1	CHAP and PA
2	CHAP only
3	PAP only
4	EAP only
5	EAP and PAP

6  
7

EAP and CHAP  
EAP, CHAP, and PAP

**Remote user ID required for logon.** Remote User ID authentication is required for logon to the local system. Valid values are:

0 No  
1 Yes

**Remote user ID validation method.** The method for validation of the remote user ID. Valid values are:

0 N/A  
1 Validation list  
2 Radius

**Remote user ID validation list name.** The name of the Validation list containing the remote User ID and password for authenticating the connection with the remote system.

**Requested PPPoE server name.** The PPPoE server name requested for this PPPoE initiator profile to negotiate the remote end of the connection.

**Requested PPPoE service.** The PPPoE service requested for this PPPoE initiator profile to negotiate with the remote end of the connection.

**Request VJ header compression.** Whether VJ header compression should be performed on IP datagrams. Valid values are:

0 No  
1 Yes

**Requires IP security protection .** Whether IP security is required for connections established by this profile connection job. Valid values are:

0 No  
1 Yes

**Routing definition.** The additional routing requested when activating this profile connection job. Valid values are:

0 Not Used  
1 Add default route  
2 Additional static routes defined

**Subsystem description.** The name of the subsystem description in which the connection jobs for this profile connection job will be run.

**Time to wait (in seconds) for adding a link.** The time (in seconds) to wait before adding an additional link after the connection utilization has exceeded the specified percentage. Valid values are:

5 - 3600 (in increments of 5, default = 15)

**Time to wait (in seconds) for dropping a link.** The time (in seconds) to wait before dropping a link after the connection utilization has receded below the specified percentage. Valid values are:

**Use filter rule.** Whether a filter rule should be used by the profile connection job. Valid values are:

0	No
1	Yes

**Use Radius for connection auditing and accounting.** Whether Radius should be used for connection auditing and accounting. Valid values are:

0	No
1	Yes

## Connection Profile Detailed Parameters

The following data is returned for each profile detailed connection parameter entry, describing one connection for a profile. Multiple connection profiles may have one entry for each connection. For detailed descriptions of the fields in the table, see "Field Descriptions" on page 209.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Connection status
4	4	BINARY(4)	Maximum transmission unit (MTU)
8	8	BINARY(4)	Maximum links per multilink connection bundle
12	C	BINARY(4)	Number of active links
16	10	BINARY(4)	Line inactivity timeout
20	14	CHAR(4)	Reserved
24	18	CHAR(6)	Job number
30	1E	CHAR(10)	Job user
40	28	CHAR(10)	Job name
50	32	CHAR(10)	Line name
60	3C	CHAR(15)	Active local IP address (set when profile is active)
75	4B	CHAR(15)	Active remote IP address (set when profile is active)
90	5A	CHAR(6)	Reserved
96	60	CHAR(48)	Remote user name
144	90	CHAR(64)	Group access policy
208	D0	CHAR(32)	Filter rule name
240	F0	CHAR(1)	IP forwardin
241	F1	CHAR(1)	Proxy ARP routing
242	F2	CHAR(1)	TCP/IP header compression
243	F3	CHAR(1)	Full masquerading
244	F4	CHAR(1)	Authentication protocol
245	F5	CHAR(1)	Multilink protocol enabled
246	F6	CHAR(1)	Multilink bandwidth utilization monitoring enabled
247	F7	CHAR(1)	Reserved
248	F8	BINARY(4)	Detailed connection status
252	FC	CHAR(4)	Reserved



## Field Descriptions

**Active (binary) local IP address.** The binary local IP address of the connection established by this profile connection job.

**Active (binary) remote IP address.** The binary remote IP address of the connection established by this profile connection job.

**Authentication protocol.** The authentication protocol that was negotiated for this profile connection. Valid values are:

0	N/A
1	CHAP and PAP
2	CHAP only
3	PAP only
4	EAP only
5	EAP and PAP
6	EAP and CHAP
7	EAP, CHAP, and PAP

**Connection status.** The current status of this profile connection. Valid values are:

0	N/A
1	Inactive or ended
2	Ending
3	Starting
4	Waiting for connection
5	Connecting
6	Active

**Detailed connection status.** Additional detail of the current status of this profile connection. Valid values are:

0	No status set
256	Undefined
257	Connection operational
258	Initializing connection to modem
259	Initializing connection data structures
260	Selecting a line from a line pool
261	Requesting a shared line from current owner
262	Waiting for shared line to be available
263	Initializing modem
264	Incoming call detected
265	Dial on-demand connection requested
266	Waiting for modem to connect
267	Redialing remote system
268	Modem connected
269	Modem disconnected
270	Authenticating remote user
271	Negotiating IP address
272	Activating IP address
273	Modem or resource failure
274	Connection profile settings failure
275	Authentication failure
276	Modem failure
277	Retry threshold failure
278	Remote phone number busy
279	No local dial tone detected

280	Remote modem did not answer
281	IP address activation failure
282	PPP protocol rejected
283	PPP connection inactivity timeout
300	Sent PPPoE initiation packet
301	Received PPPoE offer from peer
302	Sent PPPoE request packet to peer
303	Received PPPoE session-confirmation from peer
304	Sent PPPoE termination packet to peer
350	Received PPPoE termination from peer
351	No response from PPPoE peer
352	PPPoE peer response did not match request sent
353	Received error from PPPoE peer
354	Unable to open communication stream
355	Unable to send packet to PPPoE peer
356	Unable to convert packet data
357	PPPoE link error
400	Starting L2TP tunnel negotiation
401	L2TP tunnel negotiation in progress
402	L2TP tunnel established
403	Starting L2TP call negotiation
404	Starting L2TP remote call negotiation
405	L2TP call established
450	L2TP tunnel authentication failed
451	L2TP tunnel maximum connections exceeded
452	Sent stop L2TP tunnel message to peer
453	Received stop L2TP tunnel message from peer
454	L2TP call maximum connections exceeded
455	Sent stop L2TP call message to peer
456	Received stop L2TP call message from peer

**Filter rule name.** The name of the filter rule that is in effect for this profile connection. A value \*NONE means that no filter rule is in use.

**Full masquerading.** Whether full masquerading is in effect for this profile connection. Valid values are:

0	No
1	Yes

**Group access policy.** The name of the group access policy that is in effect for this profile connection. A value \*NONE means that no group policy is in use.

**IP forwarding.** Whether IP forwarding is active for this profile connection. Valid values are:

0	No
1	Yes

**Job name.** The job name of this profile connection job.

**Job number.** The job number of this profile connection job.

**Job user.** The job user name of this profile connection job.

**Line name.** The name of the line description used for this profile connection.

**Line inactivity timeout.** The value used for the inactivity timeout in the line description. Valid values are 15 - 65535 seconds.

**Maximum links per multilink connection bundle.** The maximum number of links allowed per bundle for multilink connections for this profile.

**Maximum transmission unit.** The maximum size of IP datagrams that can be sent over connections started by this profile connection job. This value is valid only when the profile is active.

**Multilink protocol enabled.** Whether multilink connections are allowed for this connection profile. Valid values are:

0 No  
1 Yes

**Multilink bandwidth utilization monitoring enabled.** Whether bandwidth utilization monitoring is enabled for this profile connection. Valid values are:

0 No  
1 Yes

**Number of active links.** The number of active links that constitute this profile connection.

**Proxy ARP routing.** Whether proxy ARP routing is in effect for this profile connection. Valid values are:

0 No  
1 Yes

**Remote user name.** The name of the connected remote user that was authenticated for this profile connection. This value is valid only when authentication is enabled for this connection profile.

**TCP/IP header compression.** Whether TCP/IP header compression will be performed for this profile connection. Valid values are:

0 No  
1 Yes

## Remote Phone Numbers

The following data is returned for each connection profile remote phone number entry. Multilink connection profiles may have one entry for each connection in the link. Single connection profiles may have more than one entry to provide backup phone numbers when the primary (first) number is unavailable. For detailed descriptions of the fields in the table, see "Field Descriptions."

Offset		Type	Field
Dec	Hex		
0	0	CHAR(48)	Remote phone number
48	30	CHAR(16)	Reserved

## Field Descriptions

**Remote phone number.** A phone number that will be used to attempt a switched connection with a remote system. Valid for Dial profiles only.

**Reserved.** An ignored field.

## Error Messages

Message ID	Error Message Text
TCP8211 E	Point-to-point profile &1 not found
CPF24B4 E	Severe error while addressing parameter list.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF9872 E	Program or service program &1 in library &2 ended, reason code &3.

API introduced: V5R1

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

---

## Retrieve TCP/IP Attributes (QtocRtvTCPA) API

Required Parameter Group:	
1	Receiver variable
<b>Output</b>	Char(*)
2	Length of receiver variable
<b>Input</b>	Binary(4)
3	Format name
<b>Input</b>	Char(8)
4	Error code
<b>I/O</b>	Char(*)
Service Program: QTOCNETSTS	
Threadsafe: Yes	

The Retrieve TCP/IP Attributes (QtocRtvTCPA) API retrieves TCP/IPv4 and TCP/IPv6 stack attributes.

## Authorities and Locks

None.

## Required Parameter Group

### Receiver variable

OUTPUT; CHAR(\*)

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

### Length of receiver variable

OUTPUT; BINARY(4)

The length of the receiver variable. If this value is larger than the actual size of the receiver variable, the result may not be predictable. The minimum length is 8 bytes.

## Format name

INPUT; CHAR(8)

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

<i>TCPA0100</i>	TCP/IPv4 stack status. Refer to “TCPA0100 Format” for details on the format.
<i>TCPA0200</i>	TCP/IPv4 stack attributes in addition to TCP/IPv4 stack status. Refer to “TCPA0100 Format” and “TCPA0200 Format” on page 216 for details on the format.
<i>TCPA0300</i>	TCP/IP domain attributes in addition to TCP/IPv4 stack status. Refer to “TCPA0100 Format” and “TCPA0300 Format” on page 221 for details on the format.
<i>TCPA1100</i>	TCP/IPv6 stack status. Refer to “TCPA1100 Format” on page 223 for details on the format.
<i>TCPA1200</i>	TCP/IPv6 stack attributes in addition to TCP/IPv6 stack status. Refer to “TCPA1100 Format” on page 223 and “TCPA1200 Format” on page 224 for details on the format.

## Error code

I/O; CHAR(\*)

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

## Format of TCP/IP Attributes Information

To retrieve the current TCP/IPv4 stack status, use format “TCPA0100 Format.”

For detailed TCP/IPv4 stack attributes in addition to the TCP/IPv4 stack status, use format “TCPA0200 Format” on page 216.

For domain name system information in addition to the TCP/IPv4 stack status, use format “TCPA0300 Format” on page 221.

To retrieve the current TCP/IPv6 stack status, use format “TCPA1100 Format” on page 223.

For detailed TCP/IPv6 stack attributes in addition to the TCP/IPv6 stack status, use format “TCPA1200 Format” on page 224.

## TCPA0100 Format

This format returns information regarding the status of the TCP/IPv4 stack. For detailed descriptions of the fields in the table, see “Field Descriptions” on page 214.

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	TCP/IPv4 stack status
12	C	BINARY(4)	How long active
16	10	CHAR(8)	When last started - date
24	18	CHAR(6)	When last started - time
30	1E	CHAR(8)	When last ended - date
38	26	CHAR(6)	When last ended - time
44	2C	CHAR(10)	Who last started - job name
54	36	CHAR(10)	Who last started - job user name
64	40	CHAR(6)	Who last started - job number
70	46	CHAR(16)	Who last started - internal job identifier
86	56	CHAR(10)	Who last ended - job name
96	60	CHAR(10)	Who last ended - job user name

Offset		Type	Field
Dec	Hex		
106	6A	CHAR(6)	Who last ended - job number
112	70	CHAR(16)	Who last ended - internal job identifier
128	80	BINARY(4)	Offset to additional information
132	84	BINARY(4)	Length of additional information
136	88	BINARY(4)	Limited mode

## Field Descriptions

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

**Bytes returned.** The number of bytes of data returned.

**How long active.** How long, in seconds, the TCP/IP stack has been active if it is active currently, or how long it was active the last time it was up if it is currently inactive.

**Length of additional information.** The length in bytes of additional information returned that is not part of format TCPA0100.

**Limited mode.** The current value of the TCP/IP Limited mode flag. TCP/IPv4 can operate while the system is in the restricted state, with limited functionality.

Possible values are:

- 0 No - The system is not currently running TCP/IPv4 in limited mode.
- 1 Yes - The system is currently running TCP/IPv4 in limited mode.

**Offset to additional information.** The offset from the beginning of the receiver variable, in bytes, to the start of the next format if a format other than TCPA0100 is requested. This field allows expansion of the basic information. A value of zero is returned if only the TCPA0100 format is requested.

**Reserved.** An ignored field.

**TCP/IPv4 stack status.** The current status of the system TCP/IPv4 stack. Possible values are:

- 0 Inactive - The TCP/IPv4 stack is not operational.
- 1 Active - The TCP/IPv4 stack is operational.
- 2 Starting - The TCP/IPv4 stack not operational, but is in the process of starting.
- 3 Ending, immediate - The TCP/IPv4 stack is operational, but is in the process of ending.
- 4 Ending, controlled - The TCP/IPv4 stack is operational, but is in the process of ending.

**When last ended - date.** The date when the TCP/IP stack was last ended. The format is YYYYMMDD, where:

YYYY Year  
MM Month  
DD Day

**When last ended - time.** The time when the TCP/IP stack was last ended. The format is HHMMSS, in 24-hour time, where:

HH	Hour
MM	Minute
SS	Second

**When last started - date.** The date when the TCP/IP stack was last started. The format is YYYYMMDD, where:

YYYY	Year
MM	Month
DD	Day

**When last started - time.** The time when the TCP/IP stack was last started. The format is HHMMSS, in 24-hour time, where:

HH	Hour
MM	Minute
SS	Second

**Who last ended - internal job identifier.** A value sent to other APIs to speed the process of locating the job on the system. Only OS/400 APIs use this identifier. This field is all NULLs if the TCP/IP stack has not been ended since the last initial program load (IPL), or if the job that ended the TCP/IP stack is no longer active.

**Who last ended - job name.** The name of the job responsible for ending the TCP/IP stack the last time it was ended. If the TCP/IP stack has not been ended since the last initial program load (IPL), this field is all NULLs.

**Who last ended - job number.** The job number responsible for ending the TCP/IP stack the last time it was ended. If the TCP/IP stack has not been ended since the last initial program load (IPL), this field is all NULLs.

**Who last ended - job user name.** The name of the user responsible for ending the TCP/IP stack the last time it was ended. If the TCP/IP stack has not been ended since the last initial program load (IPL), this field is all NULLs.

**Who last started - internal job identifier.** A value sent to other APIs to speed the process of locating the job on the system. Only OS/400 APIs use this identifier. This field is all NULLs if the TCP/IP stack has not been started since the last initial program load (IPL), or if the job that started the TCP/IP stack is no longer active.

**Who last started - job name.** The name of the job responsible for starting the TCP/IP stack the last time it was started. If the TCP/IP stack has not been started since the last initial program load (IPL), this field will be all NULLs.

**Who last started - job number.** The job number of the job responsible for starting the TCP/IP stack the last time it was started. If the TCP/IP stack has not been started since the last initial program load (IPL), this field will be all NULLs.

**Who last started - job user name.** The user name of the job responsible for starting the TCP/IP stack the last time it was started. If the TCP/IP stack has not been started since the last initial program load (IPL), this field will be all NULLs.

## TCPA0200 Format

This format returns detailed information about the TCP/IPv4 stack attributes in addition to the TCP/IPv4 stack status (format TCPA0100). For detailed descriptions of the fields in the table, see “Field Descriptions.”

Offset		Type	Field
Dec	Hex		
0	0		Returns everything from format TCPA0100
Decimal and hexadecimal offsets are reached by using the offset to additional information field in format TCPA0100.		BINARY(4)	IP datagram forwarding
		BINARY(4)	UDP checksum
		BINARY(4)	Log protocol errors
		BINARY(4)	IP source routing
		BINARY(4)	TCP urgent pointer
		BINARY(4)	IP reassembly timeout
		BINARY(4)	IP time to live
		BINARY(4)	TCP keep alive
		BINARY(4)	TCP receive buffer
		BINARY(4)	TCP send buffer
		BINARY(4)	ARP cache timeout
		BINARY(4)	MTU path discovery
		BINARY(4)	MTU discovery interval
		BINARY(4)	QoS enablement
		BINARY(4)	QoS timer resolution
		BINARY(4)	QoS data path optimization
		BINARY(4)	Dead gateway detection enablement
		BINARY(4)	Dead gateway detection interval
		BINARY(4)	TCP time wait timeout
		BINARY(4)	TCP R1 retransmission count
		BINARY(4)	TCP R2 retransmission count
		BINARY(4)	TCP minimum retransmission timeout
		BINARY(4)	TCP close connection message
		BINARY(4)	Network file cache enablement
		BINARY(4)	Network file cache timeout
		BINARY(4)	Network file cache size
		➤ BINARY(4)	Explicit congestion notification ◀◀

## Field Descriptions

**ARP cache timeout.** The ARP cache time-out value, in minutes The purpose of the time-out value is to flush out-of-date cache entries from the ARP cache.

The default ARP cache time-out interval is 5 minutes. Valid values range from 1 through 1440 minutes (24 hours).

**Dead gateway detection enablement.** Whether dead gateway detection is turned on or off. Dead gateway detection is a mechanism that involves polling all attached gateways. If no reply is received to the polls, all routes using that gateway are inactivated. Possible values are:

0 Dead gateway detection is off.



1 Dead gateway detection is on. This is the default value.

**Dead gateway detection interval.** The amount of time, in minutes, between dead gateway detection polls. When the time interval is exceeded, all attached gateways are polled to determine their availability.

The default dead gateway detection interval is 2 minutes. Valid values range from 1 through 60 minutes.



**Explicit congestion notification (ECN).** If ECN is enabled routers can notify end-nodes of congestion before queues overflow. Without ECN end-nodes can only detect congestion when packets are lost due to queues overflowing.

0 ECN is not enabled for the system. This is the default value.  
1 ECN is enabled for the system.



**IP datagram forwarding.** Whether the IP layer forwards Internet Protocol (IP) datagrams between different networks. It specifies whether the IP layer is acting as a gateway.

**Note:** IP does not forward datagrams between interfaces on the same subnet.

The OS/400 implementation of TCP/IP does not include full gateway function as defined in RFC1009. Subsets of the gateway functions are supported. One of the gateway functions supported is IP datagram forwarding capabilities. The possible values are:

0 IP datagrams are not forwarded. This is the default value.  
1 IP datagrams are forwarded.

**IP reassembly timeout.** The IP datagram reassembly time, in seconds. If this time is exceeded, a partially reassembled datagram is discarded and an ICMP time exceeded message is sent to the source host.

The default IP reassembly timeout is 10 seconds. Valid values range from 5 through 120 seconds.

**IP source routing.** Whether IP source routing currently is on or off. If IP source routing is on, it means that this system is specifying the route that outgoing IP packets take instead of allowing normal dynamic routing to take place. Some firewalls will not pass datagrams that have IP source routing switched on. The possible values are:

0 IP source routing is off.  
1 IP source routing is on. This is the default value.

**IP time to live.** The current TTL value. The IP datagram time-to-live value specifies a relative limit on the number of hops across which an IP datagram remains active. The time-to-live value acts as a hop count that is decremented by each gateway to prevent internet routing loops.

**Note:** Even though this parameter is specified as a time-to-live value, it is not used as a time value. It is used as a counter. The standard description is time to live as specified in RFCs.

**Note:** This IP datagram time-to-live value is not used for datagrams sent to an IP multicast group address. The default IP datagram time-to-live value for datagram sent to an IP multicast group is always 1, as specified by the Internet standards. Individual multicast applications may override this default using the IP\_MULTICAST\_TTL socket option.

The default time-to-live value is 64. Valid values range from 1 through 255.

**Log protocol errors.** Enables a user to log protocol errors that occur during the processing of TCP/IP data. These TCP/IP stack layer functions use this parameter to determine if they log protocol-specific errors: IP, ICMP, ARP, and NAM. TCP and UDP do not log protocol errors.

The 7004 error reference code is logged when the LOGPCLERR(\*YES) option is specified and inbound datagrams are silently discarded. Silently discarded means that an ICMP message is not returned to the originating host when a datagram is discarded because of header errors. Examples of such datagrams include those with invalid checksums and invalid destination addresses.

The error reference code is for information only. No action should be taken as a result of this error reference code. It is generated to assist with remote device or TCP/IP network problem analysis.

**Note:** These error conditions cannot be processed using an APAR.

The log protocol errors parameter should be used when error conditions require the logging of TCP/IP data, such as datagrams, to determine network problems.

The data is logged in the system error log. This error log is available through the Start System Service Tools (STRSST) command. The possible values are:

0	Protocol errors are not logged.
1	Protocol errors are logged.

**MTU discovery interval.** The amount of time, in minutes, that the TCP/IP protocol stack will cache the results of a path MTU discovery. When the time interval is exceeded, the path MTU is rediscovered.

The default path MTU discovery interval is 10 minutes. Valid values range from 5 through 40320 minutes (28 days). A special value is:

-1	*ONCE - Means that path MTUs should not be recalculated after the first discovery.
----	--

**MTU path discovery.** Whether the Path Maximum Transmission Unit (MTU) discovery function is enabled on this system.

0	MTU Path Discovery is disabled for this system.
1	MTU Path Discovery is enabled for this system. This is the default value.

**Network file cache enablement.** The current enablement status of the Network File Cache (NFC) function. The Network File Cache is used for the support of FRCA (Fast Response Cache Accelerator). FCRA dramatically improves the performance of serving non-secure static content by Web and other TCP servers.

Possible values are:

0	*NO - Network file cache is currently disabled on this system.
1	*YES - Network file cache is currently enabled on this system.

**Network file cache size.** The maximum amount of storage that may be used by the Network File Cache (NFC) for the entire system. This number is the total storage used by all TCP servers for caching files. The storage being allocated is DASD or disk and is not directly allocated from main memory. Valid values range from 10 through 100000 megabytes (100GB).

**Network file cache timeout.** The maximum amount of time in seconds that a file can be cached in the Network File Cache (NFC). This attribute ensures that a file is refreshed at a regular interval. Valid values range from 30 through 604800 seconds (one week).

Special values are:

0 \*NOMAX - Network file cache entries will not timeout.

**QoS data path optimization.** The type of data path optimization in use by Quality of Service (QoS). This field indicates the extent which QoS will batch datagrams so as to optimize performance at the risk of increasing jitter, or delay. The normal setting maximizes performance by doing more batching of datagram packets. The MinDelay setting minimizes delay by doing less batching of datagram packets and just sending them when they are ready. Possible values are:

1 \*NORMAL - Maximize performance. This setting is the default.

2 \*MINDELAY - Minimize delay.

**QoS enablement.** Whether Quality of Service (QoS), IP Type of Service (TOS), or neither of the two are in use. Possible values are:

1 \*TOS - Type of Service bytes in the IP headers are in use.

2 \*YES - QoS is in use.

3 \*NO - QoS is not in use and the Type of Service byte is not in use. This setting is the default.

**QoS timer resolution.** The Quality of Service (QoS) timer resolution value in milliseconds. This field indicates the amount of control possible over delay variations. A higher timer resolution value contributes to more jitter (delay), and a lower timer resolution uses more CPU time. The timer resolution value that can be tolerated is very dependent on the application. For example, video is highly sensitive to large delay variations. To achieve a smooth rate of flow, timers need to use small timer increments. The smaller the resolution, the smoother the data flow, but at a higher cost in terms of system overhead to manage timers.

The default QoS timer resolution is 100 milliseconds. Valid values range from 5 to 5000 milliseconds.

**TCP close connection message.** The value of the TCP close connection message attribute. The TCP close connection message attribute specifies whether abnormally closed TCP connections will be logged by messages to the QTCP message queue. TCP connections could be abnormally closed for the following reasons:

- TCP connection closed due to the 10 minute Close\_Wait time\_out.
- TCP connection closed due to the R2 retry threshold being exceeded.
- TCP connection closed due to the keep alive time-out value being exceeded.

Possible values are:

1 \*THRESHOLD - At most, one abnormally closed TCP connection message per minute will be logged. This value is the default setting.

2 \*YES - ALL abnormally closed TCP connections will be logged. Note that there are some conditions that could cause MANY closed connection messages to be logged at the same time.

3 \*NO - Abnormally closed TCP connections will not be logged.

**TCP keep alive.** The amount of time, in minutes, that TCP waits before sending out a probe to the other side of a connection. The probe is sent when the connection is otherwise idle, even when there is no data to be sent.

The transmission of keep-alive packets is controlled by individual sockets applications through use of the SO\_KEEPALIVE socket option. For more information, Sockets Programming in the iSeries Information Center.

The default keep-alive time interval is 120 minutes. Valid values range from 1 through 40320 minutes (28 days).

**TCP minimum retransmission timeout.** The current value of the configurable TCP minimum retransmission timeout attribute, in milliseconds. This attribute specifies the amount of time that TCP will wait for an acknowledgement (ACK) of a packet. When this amount of time has passed without an acknowledgement, TCP will perform the first retransmission of the packet. The default TCP minimum retransmission timeout is 250 milliseconds. Valid values range from 100 through 1000 milliseconds.

**TCP R1 retransmission count.** The R1 retransmission count value. The default value is 3. Valid values range from 1 to 15, and R1 must be less than R2.

**TCP R2 retransmission count.** The R2 retransmission count value. The default value is 16. Valid values range from 2 to 16, and R2 must be greater than R1.

**TCP receive buffer.** What to allocate for the default receive buffer size. The TCP receive window size is based on this value. Decreasing this value decreases the amount of data that the remote system can send before being read by the local application. Decreasing this value may improve performance in situations where many retransmissions occur due to the overrunning of a network adapter.

**Notes:**

1. User Datagram Protocol (UDP) does not have a configurable receive buffer size.
2. This value is also used as the default receive buffer size by IP over SNA processing.
3. Setting this parameter does not guarantee the size of the TCP receive buffer. This is the default buffer size that is used for initial TCP connection negotiations. An individual application can override this value by using the SO\_RCVBUF socket option. For more information, see Sockets Programming in the iSeries Information Center.

The default TCP receive buffer size is 8192 (8K) bytes. Valid values range from 512 through 8388608 (8MB) bytes.

**TCP send buffer.** The TCP send buffer size. This parameter informs TCP what to use for the default send buffer size. The TCP send buffer size provides a limit on the number of outgoing bytes that are buffered by TCP. Once this limit is reached, attempts to send additional bytes may result in the application blocking until the number of outgoing bytes buffered drops below this limit. The number of outgoing bytes buffered is decremented when the remote system acknowledges the data sent.

**Notes:**

1. This value is used also as the default send buffer size by IP over SNA processing.
2. UDP does not have a configurable send buffer size.
3. Setting this parameter does not guarantee the size of the TCP send buffer. This is the default buffer size that is used for initial TCP connection negotiations. An individual application can override this value by using the SO\_SNDBUF socket option. For more information, see Sockets Programming in the iSeries Information Center.

The default TCP send buffer size is 8192 (8K) bytes. Valid values range from 512 through 8388608 (8M) bytes.

**TCP time wait timeout.** The amount of time, in seconds, for which a socket pair (client IP address and port, server IP address and port) cannot be reused after a connection is closed. The maximum value possible is 2 MSL (maximum segment lifetime). The default value is 120 seconds. Valid values range from 0 (no timer) to 14400 seconds (240 minutes).

**TCP urgent pointer.** The convention to follow when interpreting which byte the urgent pointer in the TCP header points to. The urgent pointer in the TCP header points to either the byte immediately following the last byte of urgent data (BSD convention) or the last byte of the urgent data (RFC convention).

**Note:** This value must be consistent between the local and remote ends of a TCP connection. Socket applications that use this value must use it consistently between the client and server applications. This value is set on a system basis. All applications using this system will use this value. The possible values are:

- 1 Use the BSD defined convention. The TCP urgent pointer points to the byte immediately following the last byte of urgent data. This is the default value.
- 2 Use the RFC defined convention. The TCP urgent pointer points to the last byte of the urgent data.

**UDP checksum.** Whether UDP processing should generate and validate checksums. It is strongly recommended that you use UDP checksum processing. If you are concerned about obtaining the best possible performance and are not concerned with the protection provided by UDP checksum processing, turn UDP checksum processing off. The possible values are:

- 0 Checksum protection is not provided for UDP data.
- 1 Checksum protection is provided for UDP data. This is the default value.

## TCPA0300 Format

This format returns detailed information about the TCP/IP domain attributes, in addition to the TCP/IPv4 stack status (format TCPA0100). For detailed descriptions of the fields in the table, see “Field Descriptions” on page 222.

Offset		Type	Field
Dec	Hex		
0	0		Returns everything from format TCPA0100
Decimal and hexadecimal offsets are reached by using the offset to additional information field in format TCPA0100.		BINARY(4)	Offset to list of internet addresses
		BINARY(4)	Number of internet addresses
		BINARY(4)	Entry length for list of internet addresses
		BINARY(4)	DNS protocol
		BINARY(4)	Retries
		BINARY(4)	Time interval
		BINARY(4)	Search order
		BINARY(4)	Initial domain name server
		BINARY(4)	DNS listening port
		CHAR(64)	Host name
		CHAR(255)	Domain name
		CHAR(1)	Reserved
		CHAR(255)	Domain search list

**List of Internet Addresses.** These fields repeat for each Domain Name Server (DNS) Internet address.

Offset		Type	Field
Dec	Hex		
0	0	CHAR(15)	Internet address
15	F	CHAR(1)	Reserved
16	10	BINARY(4)	Internet address binary

## Field Descriptions

**DNS listening port.** The remote TCP/IP port number used to contact the Domain Name Server (DNS) or Servers listed in the Internet address parameter. 53 is the well-known port used for this purpose.

**Note:** Use of a TCP/IP port number other than the well-known port 53 for use by the Domain Name Server (DNS) can result in TCP/IP communication problems. You may inadvertently use a port number that is reserved for use by another TCP/IP application.

The default DNS Listening port is 53. Valid values range from 1 to 65532.

**DNS protocol.** The TCP/IP protocol used to communicate with the Domain Name Server (DNS) specified in the Internet address parameter. User Datagram Protocol (UDP) typically is used for this purpose. Use TCP only if your Domain Name Server (DNS) is specifically configured to use the Transmission Control Protocol (TCP). Possible values are:

- 1 Use of the User Datagram Protocol (UDP) to communicate with the Domain Name Server or Servers.
- 2 Use of the Transmission Control Protocol (TCP) to communicate with the Domain Name Server or Servers.

**Domain name.** The name of the TCP/IP domain of which this system is a member.

**Domain search list.** The TCP/IP domains to be searched whenever a host name is not given as a Fully Qualified Domain Name (FQDN). Up to six domains may be specified, separated by spaces. The list is null terminated.

**Entry length for list of internet addresses.** The entry length in bytes of each element in the list of Domain Name Server (DNS) Internet addresses returned with this format. A value of zero is returned if the list is empty.

**Host name.** The TCP/IP host name of this system. This field returns the value specified by the CHGTCPDMN command, and is the preferred system name if the system has more than one name corresponding to multiple interfaces.

**Note:** This system's TCP/IP host name must also be defined in the local host table or the Domain Name Server (DNS) specified in the Internet address parameter. If no Domain Name Server (DNS) is specified, the local TCP/IP host table is used.

**Initial domain name server.** How the initial Domain Name Server (DNS) is chosen when doing a name lookup. The first configured server can always be queried first, or TCP/IP can rotate through the configured servers in a round-robin fashion to provide a form of load balancing on the servers. Possible values are:

- 1 First. Do not rotate through the configured Domain Name Servers (DNS); always start with the first one. This setting is the default.
- 2 Rotate. Rotate through the configured Domain Name Servers (DNS) in a round-robin fashion to choose the first one to query.

**Internet address.** The IP address of a Domain Name Server (DNS) to be used by this system. There may be zero, one, two, or three Domain Name Server (DNS) Internet addresses.

If the first Domain Name Server (DNS) in the list does not respond, the second DNS server in the list will be contacted. If the second DNS server does not respond, the third DNS server will be contacted, and so on.

This field is specified in dotted-decimal form.

**Internet address binary.** The binary representation of a Domain Name Server (DNS) IP address.

**Number of internet addresses.** The number of elements in the list of Domain Name Server (DNS) Internet addresses returned with this format. A value of zero is returned if the list is empty.

**Offset to list of internet addresses.** The offset from the beginning of the receiver variable, in bytes, to the first element in the list of Domain Name Server (DNS) Internet addresses returned with this format. A value of zero is returned if the list is empty.

**Retries.** The number of additional attempts made to establish communication with each Domain Name Server (DNS), in the event the first attempt fails.

The default number of retries is 2. Valid values range from 0 to 99.

**Search order.** Whether to search a Domain Name Server (DNS) first to resolve a TCP/IP host name conflict, or to search the local TCP/IP host table first.

- 1 Local - This system will first search the TCP/IP host table, located on this system, to resolve TCP/IP host names.
- 2 Remote - This system will search a remote or local Domain Name Server (DNS) to resolve TCP/IP host names before searching the local TCP/IP host table. The Domain Name Server (DNS) to use is specified by the Internet Address parameter. This is the default value.

**Time interval.** The length of time in seconds this system will wait before initiating a retry attempt to connect to a DNS server. The default time interval is 2 seconds. Valid values range from 0 to 99.

## TCPA1100 Format

This format returns information regarding the status of the TCP/IP<sub>v6</sub> stack. For detailed descriptions of the fields in the table, see "Field Descriptions."

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	TCP/IP <sub>v6</sub> stack status
12	C	BINARY(4)	Offset to additional information
16	10	BINARY(4)	Length of additional information

## Field Descriptions

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

**Bytes returned.** The number of bytes of data returned.

**Length of additional information.** The length in bytes of additional information returned that is not part of format TCPA1100.

**Offset to additional information.** The offset from the beginning of the receiver variable, in bytes, to the start of the next format if format TCPA1200 is requested. This field allows expansion of the basic information. A value of zero is returned if only the TCPA1100 format is requested.

**TCP/IPv6 stack status.** The current status of the system TCP/IPv6 stack. Possible values are:

- 0 Inactive - The TCP/IPv6 stack is not operational.
- 1 Active - The TCP/IPv6 stack is operational.
- 2 Starting - The TCP/IPv6 stack not operational, but is in the process of starting.
- 3 Ending, immediate - The TCP/IPv6 stack is operational, but is in the process of ending.
- 4 Ending, controlled - The TCP/IPv6 stack is operational, but is in the process of ending.

## TCPA1200 Format

This format returns detailed information about the TCP/IPv6 stack attributes in addition to the TCP/IPv6 stack status (format TCPA1100). For detailed descriptions of the fields in the table, see “Field Descriptions.”

Offset		Type	Field
Dec	Hex		
0	0		Returns everything from format TCPA1100
Decimal and hexadecimal offsets are reached by using the offset to additional information field in format TCPA1100.		BINARY(4)	ICMP error message send rate time
		BINARY(4)	Router solicitation max delay
		BINARY(4)	Router solicitation interval
		BINARY(4)	Router solicitation max transmits
		BINARY(4)	Neighbor advertisement max transmits
		BINARY(4)	Neighbor solicitation delay first probe time
		BINARY(4)	Neighbor solicitation max unicast solicits
		BINARY(4)	Neighbor solicitation max multicast solicits
		BINARY(4)	TCP keep alive
		BINARY(4)	TCP urgent pointer
		BINARY(4)	TCP receive buffer size
		BINARY(4)	TCP send buffer size
		BINARY(4)	TCP R1 retransmission count
		BINARY(4)	TCP R2 retransmission count
		BINARY(4)	TCP closed timewait timeout
	BINARY(4)	TCP minimum retransmission timeout	

## Field Descriptions

**ICMP error message send rate time.** The current value of the ICMP error message send rate time attribute, in milliseconds. The ICMP error message send rate time attribute controls how often ICMPv6 error messages will be sent out by the system. This control mechanism allows the bandwidth and forwarding costs of sending ICMPv6 error messages to be limited, as in the case of many ICMPv6 error messages being generated in response to another host sending a stream of erroneous packets. The default ICMP error message send rate time is 1000 milliseconds (1 second). Valid values range from 10 through 5000 milliseconds (5 seconds).



**Neighbor advertisement max transmits.** The current value of the TCP/IPv6 stack Neighbor advertisement max transmits attribute. The Neighbor advertisement max transmits attribute is specified as a number of transmissions, and is the maximum number of unsolicited Neighbor Advertisements that the system will send at a time. The system might send unsolicited Neighbor Advertisements when one of its link-layer addresses changes (for example, hot-swap of a physical interface card). The default value of the Neighbor advertisement max transmits attribute is 3 transmissions. Valid values range from 1 through 5 transmissions.

**Neighbor solicitation delay first probe time.** The current value of the configured Neighbor solicitation delay first probe time attribute. This attribute controls how long a Neighbor Cache entry will stay in the DELAY state before the stack will send another Neighbor Solicitation and move the Neighbor Cache entry's Reachability state to PROBE if reachability still has not been confirmed. The default Neighbor solicitation delay first probe time is 5 seconds. Valid values range from 3 through 10 seconds.

**Neighbor solicitation max multicast solicits.** The current value of the configured Neighbor solicitation max multicast solicits stack attribute. This attribute controls the maximum number of multicast Neighbor Solicitations which will be sent out when the system is performing link-layer address resolution for another host (neighbor). If no Neighbor Advertisement is received after the maximum number of Neighbor Solicitations have been sent out, address resolution has failed, and an ICMPv6 error message will be returned to the application. The default value of the Neighbor solicitation max multicast solicits attribute is 3 transmissions. Valid values range from 1 through 5 transmissions.

**Neighbor solicitation max unicast solicits.** The current value of the configured Neighbor solicitation max unicast solicits stack attribute. This attribute controls the maximum number of unicast Neighbor Solicitations which will be sent out when the system is performing link-layer address resolution for another host with unicast Neighbor Solicitations. Multicast is the normal way to perform Neighbor Discovery, but unicast Neighbor Solicitations will be used if the local physical interface is not multicast-capable. If no Neighbor Advertisement is received after the maximum number of Neighbor Solicitations have been sent out, address resolution has failed, and an ICMPv6 error message will be returned to the application. The default Neighbor solicitation max unicast solicits value is 3 transmissions. Valid values range from 1 through 5 transmissions.

**Router solicitation interval.** The Router solicitation interval is the amount of time, in seconds, to wait between sending Router Solicitations while waiting for a Router Advertisement in reply. The default Router solicitation interval is 4 seconds. Valid values range from 2 through 5 seconds.

**Router solicitation max delay.** The Router solicitation max delay attribute is the amount of time, in milliseconds, to wait for a Router Advertisement reply after sending the last Router Solicitation. This attribute is also used to calculate when to send the first Router Solicitation. To avoid congestion on a link when many hosts start up at the same time (such as after a power failure), the system will wait Router solicitation max delay seconds before sending the first Router Solicitation. The default Router solicitation max delay is 1000 milliseconds. Valid values range from 500 through 3000 milliseconds.

**Router solicitation max transmits.** The maximum number of Router Solicitations to transmit. If no Router Advertisements are received in response to the transmitted Router Solicitations, the system concludes that there is no IPv6 router on its link. The default Router solicitation max transmits value is 3 transmissions. Valid values range from 1 through 5 transmissions.

**TCP closed timewait timeout.** The amount of time, in seconds, for which a socket pair (client IP address and port, server IP address and port) cannot be reused after a connection is closed. The maximum value possible is 2 MSL (maximum segment lifetime). The default value is 120 seconds. Valid values range from 0 (no timer) to 14400 seconds (240 minutes).

**TCP keep alive.** The amount of time, in minutes, that TCP waits before sending out a probe to the other side of a connection. The probe is sent when the connection is otherwise idle, even when there is no data to be sent.

The transmission of keep-alive packets is controlled by individual sockets applications through use of the SO\_KEEPALIVE socket option. For more information, Sockets Programming in the iSeries Information Center.

The default keep-alive time interval is 120 minutes. Valid values range from 1 through 40320 minutes (28 days).

**TCP minimum retransmission timeout.** The current value of the configurable TCP minimum retransmission timeout attribute, in milliseconds. This attribute specifies the amount of time that TCP will wait for an acknowledgement (ACK) of a packet. When this amount of time has passed without an acknowledgement, TCP will perform the first retransmission of the packet. The default TCP minimum retransmission timeout is 250 milliseconds. Valid values range from 100 through 1000 milliseconds.

**TCP R1 retransmission count.** The R1 retransmission count value. The default value is 3. Valid values range from 1 to 15, and R1 must be less than R2.

**TCP R2 retransmission count.** The R2 retransmission count value. The default value is 16. Valid values range from 2 to 16, and R2 must be greater than R1.

**TCP receive buffer size.** The TCP receive buffer size in bytes. The TCP receive window size is based on this value. Decreasing this value decreases the amount of data that the remote system can send before being read by the local application. Decreasing this value may improve performance in situations where many retransmissions occur due to the overrunning of a network adapter.

**Notes:**

1. User Datagram Protocol (UDP) does not have a configurable receive buffer size.
2. This value is also used as the default receive buffer size by IP over SNA processing.
3. Setting this parameter does not guarantee the size of the TCP receive buffer. This is the default buffer size that is used for initial TCP connection negotiations. An individual application can override this value by using the SO\_RCVBUF socket option. For more information, see Sockets Programming in the iSeries Information Center.

The default TCP receive buffer size is 8192 (8K) bytes. Valid values range from 512 through 8388608 (8MB) bytes.

**TCP send buffer size.** The TCP send buffer size in bytes. This parameter informs TCP what to use for the default send buffer size. The TCP send buffer size provides a limit on the number of outgoing bytes that are buffered by TCP. Once this limit is reached, attempts to send additional bytes may result in the application blocking until the number of outgoing bytes buffered drops below this limit. The number of outgoing bytes buffered is decremented when the remote system acknowledges the data sent.

**Notes:**

1. This value is used also as the default send buffer size by IP over SNA processing.
2. UDP does not have a configurable send buffer size.
3. Setting this parameter does not guarantee the size of the TCP send buffer. This is the default buffer size that is used for initial TCP connection negotiations. An individual application can override this value by using the SO\_SNDBUF socket option. For more information, see Sockets Programming in the iSeries Information Center.

The default TCP send buffer size is 8192 (8K) bytes. Valid values range from 512 through 8388608 (8M) bytes.

**TCP urgent pointer.** The convention to follow when interpreting which byte the urgent pointer in the TCP header points to. The urgent pointer in the TCP header points to either the byte immediately following the last byte of urgent data (BSD convention) or the last byte of the urgent data (RFC convention).

**Note:** This value must be consistent between the local and remote ends of a TCP connection. Socket applications that use this value must use it consistently between the client and server applications. This value is set on a system basis. All applications using this system will use this value. The possible values are:

- 1 Use the BSD defined convention. The TCP urgent pointer points to the byte immediately following the last byte of urgent data. This is the default value.
- 2 Use the RFC defined convention. The TCP urgent pointer points to the last byte of the urgent data.

## Error Messages

Message ID	Error Message Text
TCP84C6 E	Internal operations error - RESULT &1 CC &2 RC &3 ERRNO &4.
CPF24B4 E	Severe error while addressing parameter list.
CPF3C19 E	Error occurred with receiver variable specified.
CPF3C21 E	Format name &1 is not valid.
CPF3C24 E	Length of the receiver variable is not valid.
CPF3C90 E	Literal value cannot be changed.
CPF3CF1 E	Error code parameter not valid.
CPF8100 E	All CPF81xx messages could be returned. xx is from 01 to FF.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

API introduced: V5R1

[Top](#) | [“Communications APIs,”](#) on page 1 | [APIs by category](#)

## Update DNS API (QTOBUPDT)

Required Parameter Group:	
1	Update instructions
<b>Input</b>	Char(*)
2	Length of update instructions
<b>Input</b>	Binary(4)
3	Format name of update instructions
<b>Input</b>	Char(8)
4	Update key override
<b>Input</b>	Char(*)
5	Length of update key override
<b>Input</b>	Binary(4)
6	Format name of update key override
<b>Input</b>	Char(8)
7	Update key name
<b>Input</b>	Char(*)
8	Length of update key name
<b>Input</b>	Binary(4)
9	IP address of DNS server
<b>Input</b>	Char(15)
10	Miscellaneous attributes
<b>Input</b>	Char(*)
11	Length of miscellaneous attributes
<b>Input</b>	Binary(4)
12	Format name of miscellaneous attributes
<b>Input</b>	Char(8)
13	Result code
<b>Output</b>	Binary(4)
14	Error code
<b>I/O</b>	Char(*)
Program Name: QDNS/QTOBUPDT	
Default Public Authority: *USE	
Threadsafe: No	

The **Update DNS API (QTOBUPDT)** allows the caller to send one or more update instructions to an iSeries dynamic DNS (Domain Name System) server. The instructions allow for adding or deleting DNS Resource Records (RRs). The instructions can optionally include any number of prerequisite conditions that must be true for the actual updates to take place. This API is based on the Berkeley Internet Name

Domain (BIND) version 8.2.x implementation of dynamic DNS updates. Therefore, it also can be used to send update requests to DNS servers running on other operating system platforms that conform to BIND Version 8 update protocols.

**OS/400 Option 31 (Domain Name System)** must be installed to use this API.

## Authorities and Locks

If an Integrated File System (IFS) stream file name is specified for any of the parameters that allow it, then the user will need \*R authority to the stream file and \*X authority to the directories in the path of the stream file.

## Required Parameter Group

### Update instructions

INPUT; CHAR(\*)

One or more instructions that define which DNS resource records should be updated (added or deleted) for a specific DNS domain, as well as any prerequisites that must be true for those updates to take place. Depending on which format name for this parameter is chosen, this parameter will either contain the actual update instructions themselves or the name of an Integrated File System file that contains the update instructions.

The syntax for the update instructions themselves is the same as that defined by BIND 8.2.3 for dynamic DNS updates, which it uses as input to its **nsupdate** program. Please see “Update Instructions Syntax” on page 231 for descriptions of the update instructions themselves.

### Length of update instructions

INPUT; BINARY(4)

The length of the data passed in the Update instructions parameter. If the length is larger than the size of the Update instructions parameter, the results may not be predictable.

### Format name of update instructions

INPUT; CHAR(8)

The format of the data being passed in the Update instructions parameter.

*DNSU0100*

Data passed represents the actual data the API should use.

*DNSU0200*

Data passed represents the path name of an Integrated File System file that contains the data the API should use.

*DNSU0300*

Data passed represents the name of a file that contains the data the API should use. The file name is in an OS/400 API path name structure. For the format of this structure, see Path name format.

### Update key override

INPUT; CHAR(\*)

This API automatically searches the default DNS dynamic update directory /QIBM/UserData/OS400/DNS/\_DYN for a dynamic update transaction signature (TSIG) key for the specific domain being updated. The caller can override the default logic and provide a transaction signature key directly to the API by using this Update key override parameter. Depending on which format name for this parameter is chosen, this parameter will either contain the actual key itself or the path name of an Integrated File System file that contains the key.

### Length of update key override

INPUT; BINARY(4)

The length of the data passed in the Update key override parameter. If the length is larger than the size of the Update key override parameter, the results may not be predictable.

**Format name of update key override**

INPUT; CHAR(8)

The format of the data being passed in the Update key override parameter.

*DNSU0100*

Data passed represents the actual data the API should use.

*DNSU0200*

Data passed represents the path name of an Integrated File System (IFS) file that contains the data the API should use.

*DNSU0300*

Data passed represents the name of a file that contains the data the API should use. The file name is in an OS/400 API path name structure. For the format of this structure, see Path name format.

**Update key name**

INPUT; CHAR(\*)

If the caller is providing a transaction signature key in the update key override parameter, then the update key name parameter must contain the name of the update key.

**Length of update key name**

INPUT; BINARY(4)

The length of the data passed in the Update key name parameter. If the length is larger than the size of the Update key name parameter, the results may not be predictable.

**IP address of DNS server**

INPUT; CHAR(15)

The IP address, in dotted decimal form, of the DNS server where the API should start searching for the primary master DNS server for the zone being updated. The parameter must be right padded with blanks if the data does not take up the entire length.

If this parameter is all blanks on input, the API will automatically search the network to determine where the primary master DNS server is located for the zone that contains the domain being updated.

**Miscellaneous attributes**

INPUT; CHAR(\*)

Optional miscellaneous runtime attributes.

**Length of miscellaneous attributes**

INPUT; BINARY(4)

The length of the data passed in the Miscellaneous attributes parameter. If the length is larger than the size of the Miscellaneous attributes parameter, the results may not be predictable.

**Format name of miscellaneous attributes**

INPUT; CHAR(8)

The format of the data being passed in the Miscellaneous attributes parameter.

*DNSA0100*

Miscellaneous runtime attributes. Refer to "DNSA0100 Format" on page 232 for details on the format.

**Result code**

OUTPUT; BINARY(4)

Whether the API processed successfully or not, and if not, what type of problem was encountered. Any code that is not 0 means that the updates were not completely successful.

0

Successful.

1

Send error. The authoritative name server could not be reached.

- 2 Failed update packet. The name server has rejected the update, either because it does not support dynamic update or due to an authentication failure.
- 3 Prerequisite failure. The update was successfully received and authenticated by the name server. The prerequisites, however, prevented the update from actually being performed.

### Error code

I/O; CHAR(\*)

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

## Update Instructions Syntax

The syntax of the update instructions for the QTOBUPDT API is the same as the syntax of the update instructions that are input to the BIND (Berkeley Internet Name Domain) Version 8.2.x program known as **nsupdate**. It is a stream file-based input format that requires carriage-return(<cr>) linefeed (<lf>) characters to define distinct “lines” of input.

In addition to accepting these instructions using stream files, like **nsupdate**, the QTOBUPDT API has added the ability (by specifying format **DNSU0100**) for an application program to build the lines of input in memory and pass them directly to the API without first having to write them to a file. It is important to note, however, that this method still requires that you build the input lines exactly as you would if you were going to write them to a stream file; that is, separated by the same <cr><lf> characters that are described below.

QTOBUPDT reads input records, one per line, each line contributing a resource record directive to a single update request. As described below, the directives can be either *prerequisite* checks or actual resource record (RR) data *update* directives. All domain names used in an update request must belong to the same DNS zone. A blank line causes the accumulated records to be formatted into a single update request and transmitted to the zone’s authoritative name servers. Additional records may follow, which are formed into additional, completely independent, update requests for that domain. For any given call to the API, multiple update requests can be made, but each group of lines belonging to each single update request must be separated by a blank line. For the last request to be transmitted, you must remember to include a blank line as the last line of your input.

Records take one of two general forms. **Prerequisite** records specify conditions that must be satisfied before the request will be processed. **Update** records specify actual data changes to be made to the DNS database. An “*update request*” consists of zero or more prerequisites, and one or more updates. Each update request is processed atomically; that is, all prerequisites must be satisfied, then all updates are performed. If any of the prerequisites within the specific *update request* fail, the actual data update directives following them will not be attempted.

QTOBUPDT API understands the following input record formats:

<code>prereq nxdomain domain-name &lt;cr&gt;&lt;lf&gt;</code>	Requires that no RR of any type exists with name domain-name.
<code>prereq yxdomain domain-name &lt;cr&gt;&lt;lf&gt;</code>	Requires that at least one RR named domain-name must exist.
<code>prereq nxrrset domain-name [class] type &lt;cr&gt;&lt;lf&gt;</code>	Requires that no RR exists of the specified type and domain-name.
<code>prereq yxrrset domain-name [class] type [data...] &lt;cr&gt;&lt;lf&gt;</code>	Requires that a RR exists of the specified type and domain-name. If data is specified, it must match exactly.
<code>update delete domain-name [class] [type [data...]] &lt;cr&gt;&lt;lf&gt;</code>	Deletes RRs named domain-name. If type and/or data is specified, <b>only</b> completely matching records are deleted.
<code>update add domain-name ttl [class] type data... &lt;cr&gt;&lt;lf&gt;</code>	Adds a new RR with specified ttl, type, and data.

### EXAMPLES

1. The following example illustrates a set of update instructions that could be sent to the QTOBUPDT API to change an IP address by deleting any existing A records for a domain name, and then inserting a new A record. Since no prerequisites are specified, the new record will be added even if there were no existing records to delete. *The trailing blank line is required to process the request.*

- record 1: `update delete test.test.com A <cr><lf>`
- record 2: `update add test.test.com 3600 A 10.1.1.1 <cr><lf>`
- record 3: `<cr><lf>`

2. In this example, a CNAME alias is added to the database only if there are no existing A or CNAME records for the domain name.

- record 1: `prereq nxrrset www.test.com A <cr><lf>`
- record 2: `prereq nxrrset www.test.com CNAME <cr><lf>`
- record 3: `update add www.test.com 3600 CNAME test.test.com <cr><lf>`
- record 4: `<cr><lf>`

3. To accomplish both of the above independent *update requests* in a single call to the QTOBUPDT API, the update instructions submitted would be:

- record 1: `update delete test.test.com A <cr><lf>`
- record 2: `update add test.test.com 3600 A 10.1.1.1 <cr><lf>`
- record 3: `<cr><lf>`
- record 4: `prereq nxrrset www.test.com A <cr><lf>`
- record 5: `prereq nxrrset www.test.com CNAME <cr><lf>`
- record 6: `update add www.test.com 3600 CNAME test.test.com <cr><lf>`
- record 7: `<cr><lf>`

## DNSA0100 Format

The following is the format used for passing miscellaneous runtime attributes to the dynamic DNS update API. For detailed descriptions of the fields in this table, see “Field Descriptions.”

Offset		Type	Field
Dec	Hex		
0	0	B	Debug flag
4	4	B	Virtual circuit flag
8	8	C	Reserved for future use

## Field Descriptions

**Debug flag.** If set on, tells the API to create a spooled print file (QPRINT) to the caller’s userid.

- 0 Debug tracing is off (default).
- 1 Debug tracing is on.

**Reserved for future use.** A reserved field that must be set to hexadecimal zeros.

**Virtual circuit flag.** If set on, tells the API to use a TCP connection instead of the default UDP packets.

- 0 Use UDP packets to communicate with the DNS server (default).
- 1 Use TCP to communicate with the DNS server.



## Error Messages


Message ID	Error Message Text
DNS0300 E	Incorrect number of parameters passed.
DNS0301 E	The update instructions parameter was null.
DNS0302 E	The length of the update instructions parameter is incorrect.
DNS0303 E	The format name of the update instructions parameter is incorrect.
DNS0304 E	The update key override parameter is null.
DNS0305 E	The format name of the update key override parameter is incorrect.
DNS0306 E	The IP address of the DNS server parameter is incorrect.
DNS0307 E	The miscellaneous attributes parameter is null.
DNS0308 E	The format name of the miscellaneous attributes parameter is incorrect.
DNS0309 D	The transaction signature key file could not be opened.
DNS0310 E	The length of miscellaneous attributes parameter is incorrect.
DNS0311 E	The miscellaneous attributes debug flag is incorrect.
DNS0312 E	The miscellaneous attributes virtual circuit flag is incorrect.
DNS0313 E	The key name parameter is null.
DNS0314 E	The length of the key name parameter is incorrect.
DNS0315 E	The transaction signature key file could not be read.
DNS030A D	The update instructions parameter was incorrect.
DNS030B D	The dynamic DNS update failed.
DNS030C D	The dynamic DNS update partially failed.
DNS030D E	The miscellaneous attributes reserved field was not zeros.
DNS030E E	The length of the update key override parameter is incorrect.
DNS030F E	The update instructions file could not be opened.

API introduced: V5R1

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

---

## CPI Communications (CPI-C)

CPI Communications (CPI-C) provides a cross-system programming interface for applications that require program-to-program communications. [CPI-C Reference](#)  details CPI-C APIs and provides examples for designing application programs using CPI-C concepts and calls.



[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

---

## Exit Programs

These are the Exit Programs for this category.

## Trace Exit Program for Trace TCP Application command

Required Parameter Group:	
1	Trace option setting
<b>Input</b>	Char(10)
2	Application
<b>Input</b>	Char(10)
3	Error detected
<b>Output</b>	Char(10)
4	Comparison data <sup>1</sup> (page 234)
<b>Input</b>	Char(*)
QSYSINC/H member name: ESCWCHT	

**Note 1:** Valid only when watch for trace event facility is enabled (WCHMSG or WCHLICLOG parameters were specified)

The Trace TCP Application (TRCTCPAPP) command has the capability to call a user-written program, if specified in the TRCPGM parameter. This user written program will be called in the following circumstances:

For SET(\*ON), the trace program will be called:

- Before the application trace starts.
- After the communications and Licensed Internal Code (LIC) traces, if requested, start.

For SET(\*OFF), the trace program will be called:

- Before the LIC traces, if requested, end.
- After the communications trace, if requested, ends.
- After the application trace ends.

For SET(\*END), the trace program will be called:

- After the LIC and communications traces, if requested, end.
- After the application trace ends.

Also, the Trace TCP Application (TRCTCPAPP) command has the capability to watch for a specific event and end the trace when this event occurs. An event can be a message being sent to a specific message queue, history log or job log; or a LIClog. If specified in the TRCPGM parameter, the watch for trace event facility will call a user-written program at the moments specified in the Trace option setting parameter.

See the online help for more information on the TRCTCPAPP command.

### Required Parameter Group

#### Trace option setting

INPUT; CHAR(10)

The reason indicating the moment at which the user-written program was called. The possible values are:

*ON	- The collection of trace information is started, or - The watch for trace facility is starting <sup>1</sup> .
*OFF	The collection of trace information is stopped and the trace information is written to spooled printer files of the user.
*END	Tracing is ended and all trace information is deleted. No trace information output is created.
*MSGID <sup>1</sup>	A match on a message id specified on WCHMSG parameter occurred.
*LICLOG <sup>1</sup>	A match on a LIC log specified on the WCHLICLOG parameter occurred.
*CMPDATA <sup>1</sup>	The major and minor code of a LIC log matched, but the comparison data did not.
*INTVAL <sup>1</sup>	The time interval specified on TRCPGMITV parameter is elapsed.
*WCHTIMO <sup>1</sup>	The length of time to watch specified on WCHTIMO is elapsed.

### Application.

INPUT; CHAR(10)

The possible values for the "Application" parameter are the same as the values for the APP parameter on the TRCTCPAPP command. Ignored by watch for trace event facility.

### Error detected

OUTPUT; CHAR(10)

Indicates if the trace event facility should stop or continue running, or if an error on the user-written program was found. The possible values are:

- \*CONTINUE<sup>1</sup> - The trace and the watch for trace event facility.
- \*STOP<sup>1</sup> - The trace and the watch for trace event facility will be ended.
- \*ERROR - Error detected by customer trace program.

### Comparison data

INPUT; CHAR(\*)

The format of the trace information depends on the Trace option setting causing the exit program to be called. The format of the Comparison data is as follows if the Trace option setting is \*MSGID:

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of trace information
4	4	CHAR(7)	Message ID
11	B	CHAR(9)	Reserved
20	14	BINARY(4)	Offset to comparison data
24	18	BINARY(4)	Length of comparison data
28	1C	CHAR(*)	Message comparison data

The format of the Comparison data is as follows if the Trace option setting is \*LICLOG or \*CMPDATA:

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of trace information
4	4	CHAR(4)	LIC Log major code
8	8	CHAR(4)	LIC Log minor code

Offset		Type	Field
Dec	Hex		
12	C	CHAR(8)	LIC Log identifier
20	14	BINARY(4)	Offset to comparison data
24	18	BINARY(4)	Length of comparison data
28	1C	CHAR(*)	LIC Log comparison data

The format of the Comparison data is as follows if the Trace option setting is \*ON, \*INTVAL or \*WCHTIMO:

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of trace information (always 4 at this time)

## Field Descriptions

**Length of trace information.** The length of the Comparison data parameter passed to the user-written exit program.

**Length of comparison data.** The length of the user specified text to be compared against the event data.

**LIC Log identifier.** The LIC Log entry identifier of the LIC Log that occurred.

**LIC Log major code.** The major code of the LIC Log that occurred.

**LIC Log minor code.** The minor code of the LIC Log that occurred.

**LIC Log comparison data.** The user specified text string used to compare against the entry data of the watched for log entry.

**Message ID.** The identifier of the message that occurred.

**Message comparison data.** The user specified text string used to compare against the entry data of the watched for message ID.

**Offset to comparison data.** The offset to the field that holds the comparison data.



Exit program introduced: V5R3

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

## Exit Program for Watch for Trace Event

Required Parameter Group:	
<b>1</b>	Trace option setting
<b>Input</b>	Char(10)
<b>2</b>	Reserved
<b>Input</b>	Char(10)
<b>3</b>	Error detected
<b>Output</b>	Char(10)
<b>4</b>	Comparison data
<b>Input</b>	Char(*)
QSYSINC/H member name: ESCWCHT	

The Trace commands like STRCMNTRC, STRTRC, TRCINT and TRCCNN have the capability to watch for a specific event and end the trace when this event occurs. An event can be a message being sent to a specific message queue, history log or job log; or a LIClog. If specified in the TRCPGM parameter, the watch for trace event facility will call a user-written program at the moments specified in the Trace option setting parameter.

See the online help for more information on the STRCMNTRC, STRTRC, TRCINT or TRCCNN commands.

### Required Parameter Group

#### Trace option setting

INPUT; CHAR(10)

The reason indicating the moment at which the user-written program was called. The possible values are:

*ON	The watch for trace facility is starting.
*MSGID	A match on a message id specified on WCHMSG parameter occurred.
*LICLOG	A match on a LIC log specified on the WCHLICLOG parameter occurred.
*CMPDATA	The major and minor code of a LIC log matched, but the comparison data did not.
*INTVAL	The time interval specified on TRCPGMITV parameter is elapsed.
*WCHTIMO	The length of time to watch specified on WCHTIMO is elapsed.

#### Error detected

OUTPUT; CHAR(10)

Indicates if the trace event facility should stop or continue running, or if an error on the user-written program was found. The possible values are:

*CONTINUE	The trace and the watch for trace event facility will continue running
*STOP	The trace and the watch for trace event facility will be ended
*ERROR	Error detected by customer trace program.

#### Comparison data

INPUT; CHAR(\*)

The format of the trace information depends on the Trace option setting causing the exit program to be called. The format of the Comparison data is as follows if the Trace option setting is \*MSGID:

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of trace information
4	4	CHAR(7)	Message ID
11	B	CHAR(9)	Reserved
20	14	BINARY(4)	Offset to comparison data
24	18	BINARY(4)	Length of comparison data
28	1C	CHAR(*)	Message comparison data

The format of the Comparison data is as follows if the Trace option setting is \*LICLOG or \*CMPDATA:

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of trace information
4	4	CHAR(4)	LIC Log major code
8	8	CHAR(4)	LIC Log minor code
12	C	CHAR(8)	LIC Log identifier
20	14	BINARY(4)	Offset to comparison data
24	18	BINARY(4)	Length of comparison data
28	1C	CHAR(*)	LIC Log comparison data

The format of the Comparison data is as follows if the Trace option setting is \*ON, \*INTVAL or \*WCHTIMO:

Offset		Type	Field
Dec	Hex		
0	0	BINARY(4)	Length of trace information (always 4 at this time)

## Field Descriptions

**Length of trace information.** The length of the Comparison data parameter passed to the user-written exit program.

**Length of comparison data.** The length of the user specified text to be compared against the event data.

**LIC Log identifier.** The LIC Log entry identifier of the LIC Log that occurred.

**LIC Log major code.** The major code of the LIC Log that occurred.

**LIC Log minor code.** The minor code of the LIC Log that occurred.

**LIC Log comparison data.** The user specified text string used to compare against the entry data of the watched for log entry.

**Message ID.** The identifier of the message that occurred.

**Message comparison data.** The user specified text string used to compare against the entry data of the watched for message ID.

**Offset to comparison data.** The offset to the field that holds the comparison data.



Exit program introduced: V5R3

[Top](#) | [“Communications APIs,” on page 1](#) | [APIs by category](#)

---

## Concepts

These are the concepts for this category.

---

## User-Defined Communications

User-defined communications support is a set of application program interfaces (APIs) that are part of the Operating System/400<sup>(R)</sup> (OS/400<sup>(R)</sup>) licensed program. These callable routines allow customers to write their own communications protocol stacks above the iSeries<sup>(TM)</sup> data link and physical layer support. The term **user-defined communications** is used here to describe this communications protocol support. The term **application program** refers to a user-supplied communications application program.

This article defines the user-defined communications support and describes how to write protocols using the APIs. In addition, it provides two C language program examples that illustrate the use of the APIs while performing a simple file transfer between two systems attached to an X.25 packet switched network.

## Overview

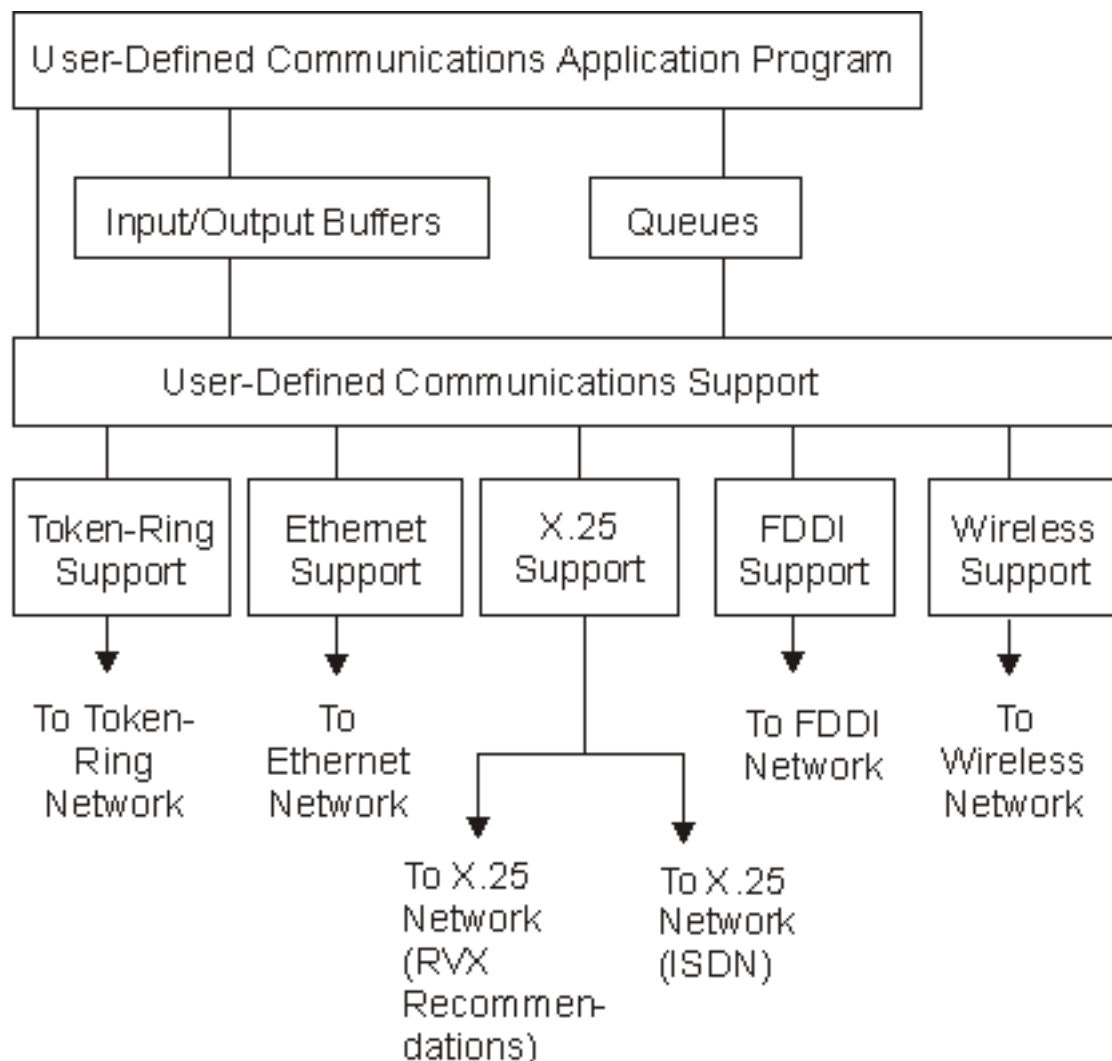
The user-defined communications APIs allow your application programs to send and receive data, and do specialized functions such as setting timers.

Your application programs need to work with the following:

- User-defined communications support
- Input/output buffers and descriptors
- A queue

Figure 1-1 (page 239) shows an overview of the user-defined communications support.

### Figure 1-1. User-Defined Communications Support



FDDI = fiber distributed data interface  
 ISDN = integrated services digital network  
 RVX = an abbreviation for the physical interface that X.25 can use:

- R = RS-232 and RS-449 (Electronic Industries Association (EIA) types)
- V = V.35 (International Telegraph and Telephone Consultative Committee (CCITT) V series types)
- X = X.21 (CCITT X series types)

## User-Defined Communications Callable Routines

The APIs provided by the OS/400 licensed program are callable routines that allow an application program to start, perform, and end communications, and perform specialized functions such as setting timers. These routines are listed below and are discussed in detail in "User-Defined Communications Support APIs" on page 1.

- Disable Link (QOLDLINK) ends communications



- Enable Link (QOLELINK) starts communications
- Query Line Description (QOLQLIND)
- Receive Data (QOLRECV)
- Send Data (QOLSEND)
- Set Filter (QOLSETF) for inbound routing information
- Set Timer (QOLTIMER) sets or cancels a timer

## Input/Output Buffers and Descriptors

The input/output buffers and descriptors are user space objects (\*USRSPC) that contain and describe the data an application program is sending or receiving. There are separate buffers and descriptors for input and output.

When an application program is ready to send data, it fills the output buffer with data and provides a description of that data in the output buffer descriptor. Similarly, when an application program receives data, the user-defined communications support fills the input buffer with data and provides a description of that data in the input buffer descriptor.

The OS/400 licensed program also provides callable APIs to allow an application program to manipulate the data in the user spaces. Some of these APIs are listed below.

- Change User Space (QUSCHGUS)
- Retrieve Pointer to User Space (QUSPTRUS)
- Retrieve User Space (QUSRTVUS)


See User Space APIs for more information.

## Queues

A queue is used by the user-defined communications support to inform an application program of some action to perform or of an activity that is complete.

The OS/400 licensed program provides APIs that allow your application programs to manipulate the data and user queues. Some of these callable APIs are listed below.

- Clear Data Queue (QCLRDTAQ)
- Create User Queue (QUSCRTUQ)
- Delete User Queue (QUSDLTUQ)
- Receive Data Queue (QRCVDTAQ)
- Send Data Queue (QSNDDTAQ)

See the CL Programming  book for more information on data queues.

---

## Terminology

Listed below are terms that are important in understanding the information contained in this part.

**Communications handle.** The name an application program assigns and uses to refer to a link.

**Connection.** The logical communication path from one computer system to another. For example, a switched virtual circuit (SVC) connection on an X.25 network.

**Connectionless service.** A method of operation where data can be sent to and received from the remote computer system without establishing a connection to it. User-defined communications support provides

connectionless service over token-ring, Ethernet, fiber distributed data interface (FDDI), wireless and X.25 networks only. For a local area network (LAN) environment, connectionless service is also known as unacknowledged service.

**Connection-oriented service.** A method of operation where a connection to the remote computer system must first be established before data can be sent to it or received from it. User-defined communications support provides connection-oriented service over X.25 networks only.

**Connection identifier.** A local identifier (ID) that a computer system uses to distinguish one connection from another. When using the user-defined communications support on the server, a connection ID is made up of a user connection end point ID and a provider connection end point ID.

**Disable.** The process of deactivating a link so that input and output operations are no longer possible on a communications line.

**Enable.** The process of setting up and activating a link for input and output operations on a communications line.

**Filter.** The technique used to route inbound data to a link that is enabled by an application program.

**Link.** The logical path between an application program and a communications line. A link is made up of the following communications objects:

- Network interface description running X.25 over ISDN
- X.25, token-ring, fiber distributed data interface (FDDI), Ethernet, or wireless line description
- Network controller description
- Network device description of type \*USRDFN

**Provider connection end point ID (PCEP ID).** The portion of the connection ID that the user-defined communications support uses to identify the connection. For example, data sent by the application program will be on the PCEP ID portion of the connection ID.

**User connection end point ID (UCEP ID).** The portion of the connection ID that the application program uses to identify the connection. For example, data received by the application program is on the UCEP ID portion of the connection ID.

---

## Relationship to Communications Standards

Figure 1-2 (page 242) shows the structure of advanced program-to-program communications (APPC) on the server and its relationship to the International Standards Organization (ISO) protocol model. Note that only the application layer above the APPC protocol code is available for definition. The APPC functional equivalents of the ISO presentation, session, networking, transport, data link, and physical layers are performed by the OS/400 operating system or Licensed Internal Code, and you cannot replace or change them. Contrast this with Figure 1-3 (page 243) which shows how much more of the protocol is defined by the user-defined communications application than by the APPC application.

**Figure 1-2. iSeries APPC versus ISO Model**

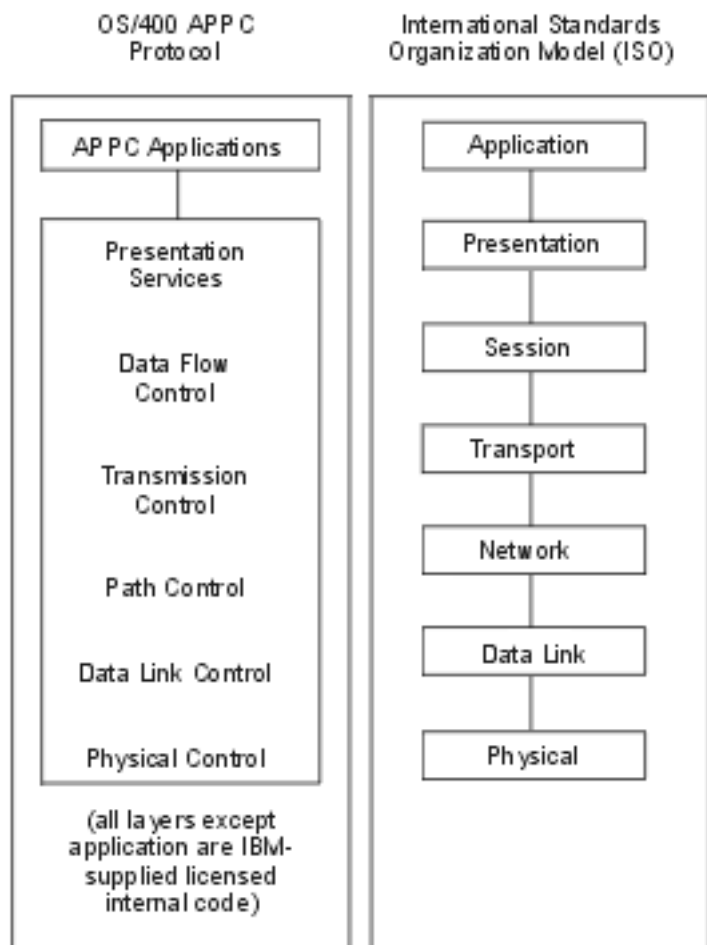
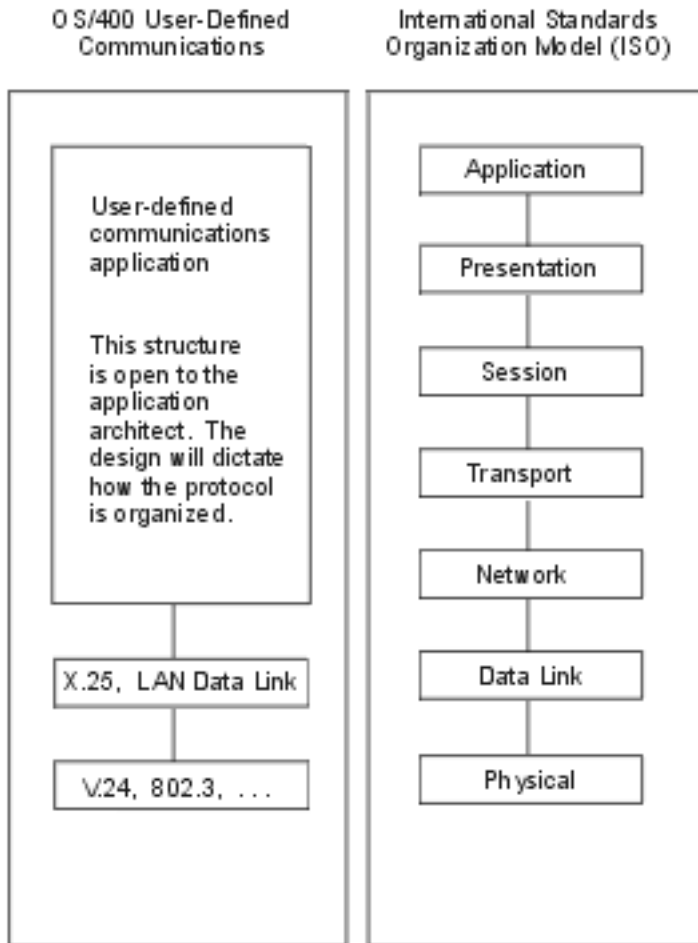


Figure 1-3 (page 243) shows the structure for user-defined communications and its relationship to the International Standards Organization (ISO) protocol model. Note that the available iSeries data links and physical layers limit user-defined communications to run over LAN (token-ring, Ethernet, wireless, or FDDI), or X.25 links, but the portion of the protocol above the data link layer is completely open to a user-defined communications application. In addition, these same X.25 and LAN links may be shared between the application program and other iSeries communications protocols that support X.25 and LAN lines. Examples include Systems Network Architecture (SNA), asynchronous communications, Transmission Control Protocol/Internet Protocol (TCP/IP), and Open Systems Interconnection (OSI).

**Figure 1-3. iSeries User-Defined versus ISO Model**



You can write protocols that run over local area networks or X.25 networks completely in high-level languages such as C, COBOL, or RPG. You can also write protocols currently running on other systems to run on the iSeries. For example, you can write both non-SNA LAN or X.25 packet layer protocols on the iSeries.

Configuration instructions also need to be supplied with the application program. User-defined communications support simply opens a pathway to the system data links. It is up to you as a protocol developer to supply any configuration instructions that are in addition to the data link or physical layer definition. Data link and physical layer definitions are defined when you use the following commands:

- Create Line Description (DDI) (CRTLINDDI)
- Create Line Description (Ethernet) (CRTLINETH)
- Create Line Description (Token Ring) (CRTLINTRN)
- Create Line Description (Wireless) (CRTLINWLS)
- Create Line Description (X.25) (CRTLINX25)
- Create Network Interface Description (ISDN) (CRTNWIISDN)

Figure 1-4 (page 244) outlines the difference between standard iSeries communications configuration, such as the iSeries APPC protocol, and user-defined communications configuration.

**Figure 1-4. Comparison between User-Defined Communications and APPC Communications**

Object	APPC Communications	User-Defined Communications
Network Interface Description	ISDN basic rate interface (BRI). Describes the physical attachment to an ISDN BRI. Only used for ISDN. X.25 or IDLC protocols supported.	Same as APPC. Only X.25 supported.
Line Description	SDLC, LAN, IDLC, X.25 lines supported. Contains local port information for iSeries communication IOP (hardware address, maximum frame size, exchange identifier (XID), local recovery information, ...).	LAN, X.25 lines supported. Same as APPC except some of the information does not apply to user-defined communications.
Controller Description	APPC, host controllers supported. Describes remote system, and parameters must match the remote hardware (hardware address, XID, ...).	Network controller supported. Pathway into network. Only one specific parameter—X.25 time-out value.
Device Description	APPC device supported. Describes remote logical unit (LU), and parameters must match partner LU (remote location name, local location name, ...).	Network device supported. Only describes the communications method or type (for example, TCP/IP, OSI, or user-defined communications).
Mode Description and Class-of-Service (COS)	Required.	Not available.

Although an APPC network requires one APPC controller description to describe each remote system in the network, user-defined communications only requires one network controller for communications with an entire network of remote systems. Thus, LAN and X.25 lines can be shared between user-defined communications support and any other protocols that support those same line types. For example, APPC may run over a token-ring line and use the X'04' Service Access Point (SAP). TCP/IP might run at the same time using the X'AA' SAP. You might write an application program to use the X'22' SAP, and run at the same time as the first two. All three protocols can be active at the same time across the same physical media.

**Note:** System-specific configuration information must be part of the application program and is not supplied by IBM<sup>(R)</sup>.

---

## Local Area Network (LAN) Considerations

User-defined communications supports these LAN types:

- Token ring (IEEE 802.5)
- Ethernet (IEEE 802.3)
- Ethernet Version 2
- Wireless
- FDDI

For token ring (802.5), Ethernet (802.3), and FDDI, user-defined communications uses the IEEE 802.2 logical link control (LLC) layer, which provides type 1 connectionless service. Connectionless service is also known as unacknowledged service. The LLC layer provides for type 2 connection service as well. For Ethernet Version 2, no 802.2 layer is available.

The wireless LAN type supports the characteristics of both Ethernet (802.3) and Ethernet Version 2.

Your application program has access to type 1 unnumbered information (UI) frames. This connectionless service is commonly referred to as *datagram* support where protocol data units are exchanged between end points without establishing a data link connection first.

The type 1 operations, test and exchange identifier (XID) frames, are not supported in user-defined communications. Any XID or test frames that the physical layer of the iSeries receives are processed by the input/output processor (IOP) and never reach your application program.


LAN frames are routed by filtering incoming data using the inbound routing data defined by your application program. The filters are hierarchical and are set up by your application program before communications is started.

The following list shows the possible settings for LAN inbound routing data (filters) from least selective to most selective.

- Destination Service Access Point (DSAP)
- DSAP, Source Service Access Point (SSAP), and optional Ethernet Version 2 frame type
- DSAP, SSAP, optional Ethernet Version 2 frame type, and adapter address

Because user-defined communications does not allow applications to define the data link and physical layers, the entire token-ring or Ethernet frame is not available to your applications. The following fields are the parts of the LAN frame that are available to the user-defined communications support:

- DSAP
- SSAP
- Destination address (DA)
- Routing information (RI)  
This field is available only when using token ring.
- Priority control  
This field is available only when using token ring.
- Access control  
This field is available only when using token ring.
- Data

For more information on local area networks, see the LAN, Frame-Relay and ATM Support  book.

---

## X.25 Considerations

X.25 user-defined communications support includes access to both permanent virtual circuits (PVCs) and switched virtual circuits (SVCs).


Over X.25 networks, including those using ISDN, your application program can initiate and accept X.25 calls, send and receive data, reset, and clear connections.

X.25 packets are routed by filtering the incoming call request using the inbound routing data that is defined by your application program. The filters are hierarchical and are set up by the application program before communications is started.

The following list shows the possible settings for X.25 inbound routing data (filters) from least selective to most selective.

- Protocol identifier (PID)
- PID, and calling data terminal equipment (DTE) address

When X.25 networks are using ISDN, notification of incoming calls may be received on the D-channel. You can decide whether these calls are accepted.

For more information on X.25 networks, see the X.25 Network Support  book.

Top | "Communications APIs," on page 1 | APIs by category

---

## Programming Design Considerations for Communications APIs

This document outlines concepts related to user-defined communications and how they might relate to the design of a user-defined communications application. Topics covered are:

- Jobs
- Application program feedback
- Programming languages
- Connection identifiers
- Token-ring, Ethernet, and wireless networks
- X.25 networks
- Queues
- User spaces


### Jobs

A fundamental concept in user-defined communications is the job. The concept of the job is important because the user-defined communications support performs services for the job requesting the communications support through one of the user-defined communications APIs. Information used by the user-defined communications support is kept along with other information about the job. You can display this information by using the Work with Job (WRKJOB) command and selecting the Work with communications status option. The user-defined communications information for the job, such as the communications handle name, last operation, and input and output counts are shown.

A user-defined communications application program (hereafter referred to as an application or application program), always runs within a job. This job may be run interactively or in batch and always represents a separate application to the user-defined communications support. This means that the same protocol can be actively running in more than one job on the system. Also, more than one job can have links that share the same line as other jobs running application programs.

Each link that is enabled by an application program logically consists of the line, network controller, and network device description objects (plus the network interface description object for ISDN links). Many applications can share the same line and controller description, provided the applications are running in different jobs, but each application uses a different device description. Up to 256 device descriptions can be attached to a controller description. This means that there can be a maximum of 256 jobs running application programs that share the same line at one time. When an application program has finished using a link and disabling it, the network device description used by the application becomes available to another application.

For end-to-end communication to begin, the application programs on each system must be started. There is no function equivalent to the intersystem communications function (ICF) program start request. Your application program is responsible for providing this support, if needed. To provide this support, your application can have a batch job servicing remote requests to start the user-defined communications application program. This job can be created to run in any subsystem.

For more information on jobs and subsystems, see the Work Management  book on the V5R1 Supplemental Manuals Web site.

You can design your application programs so that the entire protocol resides within one job or separate jobs where each job represents a portion of the protocol.

There is a one-to-one correspondence between a job and the user-defined communications support for that job. The user-defined communications support for one job does not communicate with the user-defined communications support for another job. If two applications wish to communicate between

themselves, a method such as a shared queue can be used. Also, the queue can be shared between the two (or more) jobs and the user-defined communications support for those jobs.

Figure 1-1 (page 248) shows how user-defined communications relate to the OS/400<sup>(R)</sup> job structure and the data queue or user queue that provides the ability to communicate between your application and the user-defined communications support.

In this figure, one interactive job is running over an X.25 line (X25USA) to a system in Rochester, Minnesota, using the user-defined communications support. The link was enabled with communications handle name ROCHESTER.

The user space application programming interfaces (APIs) that the application program is using are shown, along with the programming interfaces for data and user queues and the user-defined communications support APIs.

**Figure 1-1. Overview of API Relationships**

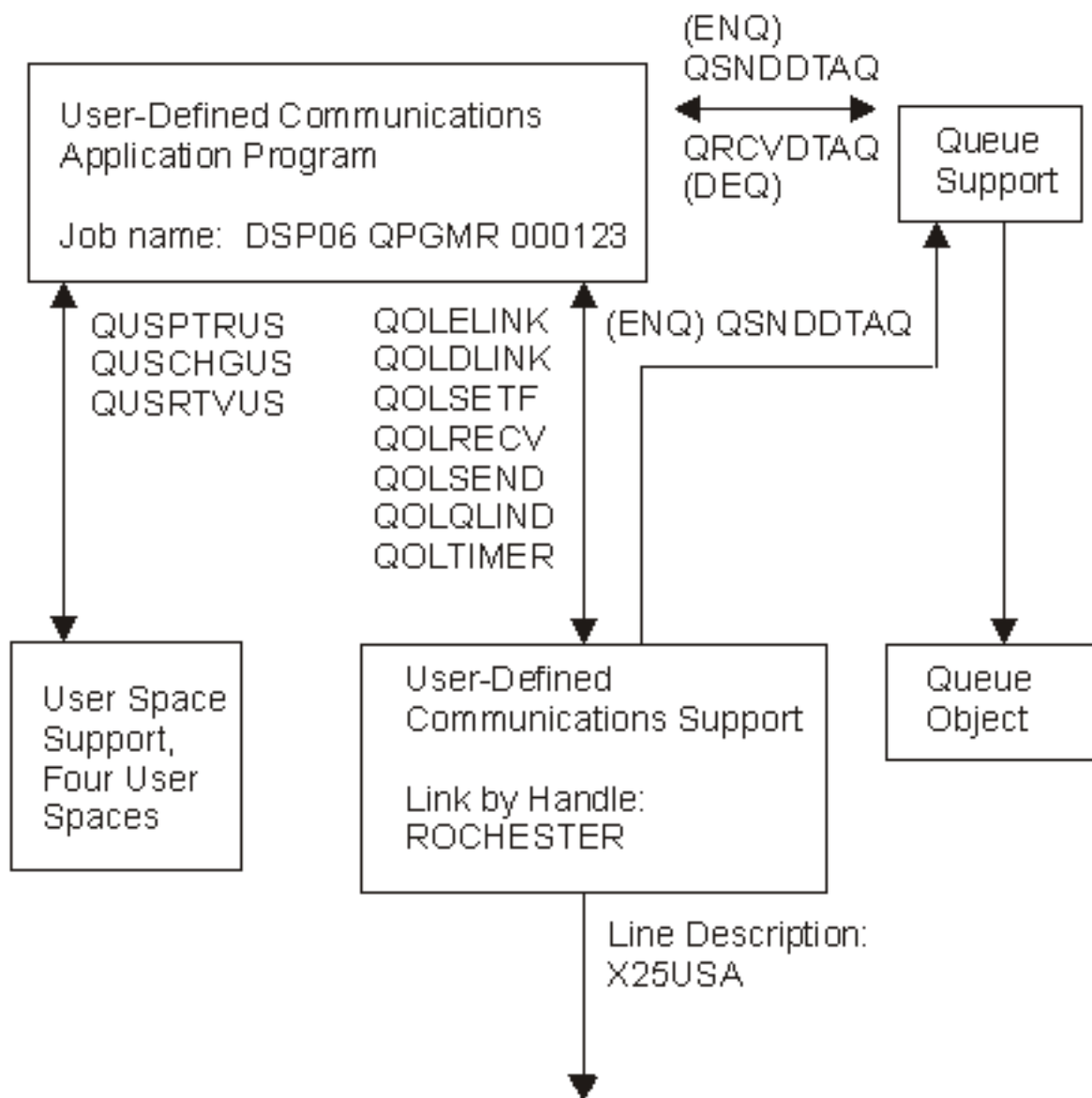
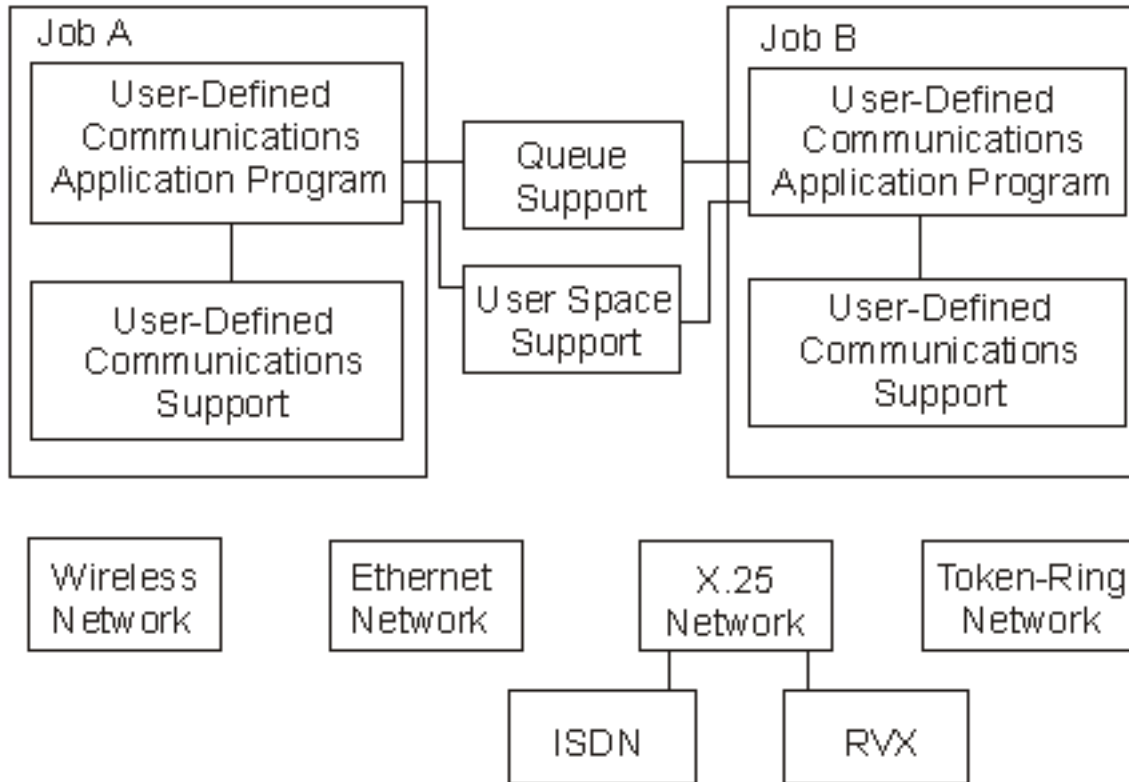




Figure 1-2 (page 249) shows two jobs, A and B. Each job is using the user-defined communications support to communicate with the networks attached to the iSeries<sup>™</sup> server by the line description. The figure shows the relationship between the different APIs and the job which is running the application program.

The lines between the jobs indicate that callable APIs that are used to communicate between the application program and the system services shown.

**Figure 1-2. Application Programming Interface to Job Structure**



The following list pertains to Figure 1-2 (page 249).

- The applications use the data queue APIs, user space APIs, and user-defined communications APIs.
- An application can have more than one link enabled, and can use a separate queue for each link, or the same queue for some or all the links that it has enabled.
- The two jobs can communicate with each other using a common queue. This queue can be the same queue that is used for user-defined communications support or a different one.
- Both jobs (or any other job on the system) that has the proper authority to the user spaces, can access the user spaces.
- The user-defined communications support uses the data in the output user spaces that are created when the link is created. The application making the call to the Send Data (qolsend) API can fill the output buffer and descriptor, or another application can do this.
- The user-defined communications support sends data to the application through the input buffer and input descriptor that is created when the link on which the data is arriving was created. Either the application making the call to the Receive Data (QOLRECV) API retrieves the data from the input buffer and descriptor, or another application with access to the user spaces does this.

- The application supplies any communications handle (link name) to the link as long as this name is unique among all the other links that the job has enabled.
- An application can enable as many links as there are line descriptions that are supported (X.25, token-ring, Ethernet, wireless, and FDDI) and that are able to be varied on.
- An application is able to run over X.25 and LAN links concurrently.

## Application Program Feedback

The user-defined communications support uses return and reason codes to indicate the success or failure of an operation, and provide suggested recovery information. In severe error conditions an escape message is signaled to the application program. If a severe error occurs, user-defined communications is no longer available to the application.

When the `qolsend` and `QOLRECV` APIs return to the application and you are running to an X.25 network, the diagnostic field is filled in. The reason code indicates whether or not the application program should look at the data returned in the diagnostic field. The diagnostic field contains additional information on the error or condition that is reported.

## Synchronous and Asynchronous Operations

Most operations that an application program requests on the call to the `qolsend` API are synchronous operations. Synchronous operations involve one step, which is to call the `qolsend` API, passing the appropriate information. Synchronous operations complete when the `qolsend` API returns to the application program. The success or failure of the operation is reported in the return and reason codes by the `qolsend` API.

Asynchronous operations do not complete when the `qolsend` API returns to the application. There are two steps for every asynchronous operation:

1. Call the `qolsend` API to initiate or request the operation.
2. Call the `QOLRECV` API to receive the results of the completed operation.

When the `qolsend` API returns to the application program, the request for the operation is successfully submitted. After the requested operation is complete, the user-defined communications support sends an incoming data entry (if necessary) to the queue to instruct the application program to call the `QOLRECV` API to receive the data. When this call to the `QOLRECV` API returns, the return and reason codes in the parameter list contain the success or failure of the operation. If the operation was unsuccessful due to an application template error in the user space used for output, the request data given to `qolsend` using the output buffer and descriptor is copied into the input buffer and descriptor. The offset to the template error detected is returned in the parameter list of the `QOLRECV` API. Asynchronous operations are only used for open connection requests, close connection requests, and resets.

For either type of operation, the application program is allowed to use the output user spaces again as soon as the call to the `qolsend` API returns.

## Programming Languages

Any program written in an OS/400-supported language can call user-defined communications support. One consideration for choosing one language over another, is that the programming language must have the ability to set a byte field to any hexadecimal value. This does not restrict programming in the different languages, but it does make some languages more appealing than others.

## Starting and Ending Communications

Relatively little configuration is required by user-defined communications support to begin communications to the network. For information on configuration, see "Configuration and Queue Entries" on page 284.

To start communications with a network, your user-defined communications application program enables the link to the network by calling the Enable Link (QOLELINK) API. Once the link is enabled, the application program can call any of the user-defined communications support APIs, and request any of the operations supported for the link. When the application program completes communications with the network, it disables the link by calling the Disable Link (QOLDLINK) API.

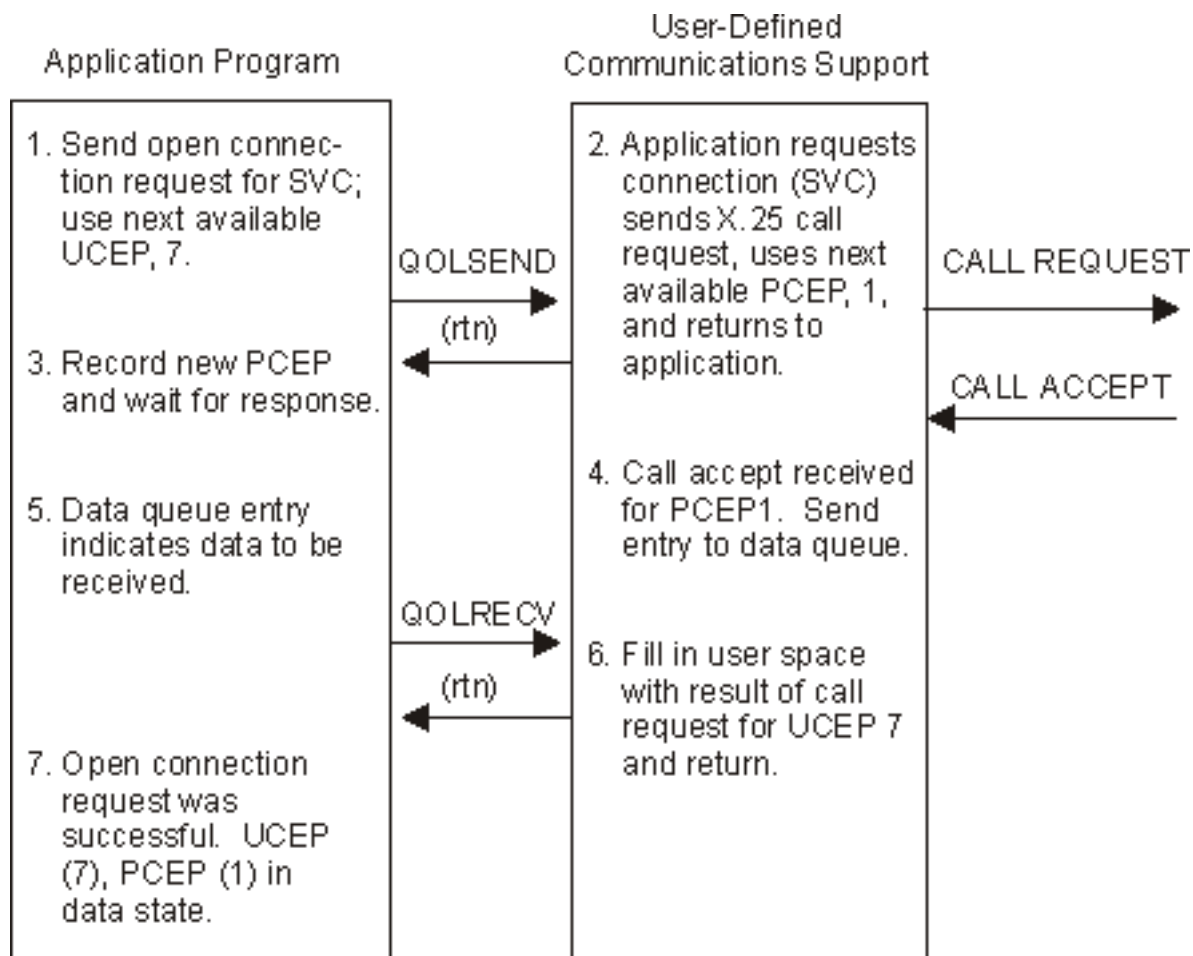
**Note:** Enabling the link does not result in any communications activity on the network. Disabling a link may cause communications activity for X.25 links if connections are active when the link is disabled.

## Using Connection Identifiers

Connection identifiers are used for connection-oriented support over X.25 networks. The connectionless connection identifiers (UCEP=1, PCEP=1) are used for local area networks. The following examples (Figure 1-3 (page 251) through Figure 1-14 (page 267)) illustrate how to use connection identifiers (UCEP and PCEP). They show how the two step operations, open connection request, and close connection request relate to the UCEP and PCEP identifiers. Note the outstanding two-step operations. This is important so that the application can correctly interpret the PCEP and reuse UCEPs.

The connections in each figure refer to SVC connections, and the examples use the Receive Data Queue (QRCVDTAQ) API. The same principles apply when using PVC connections and user queues.

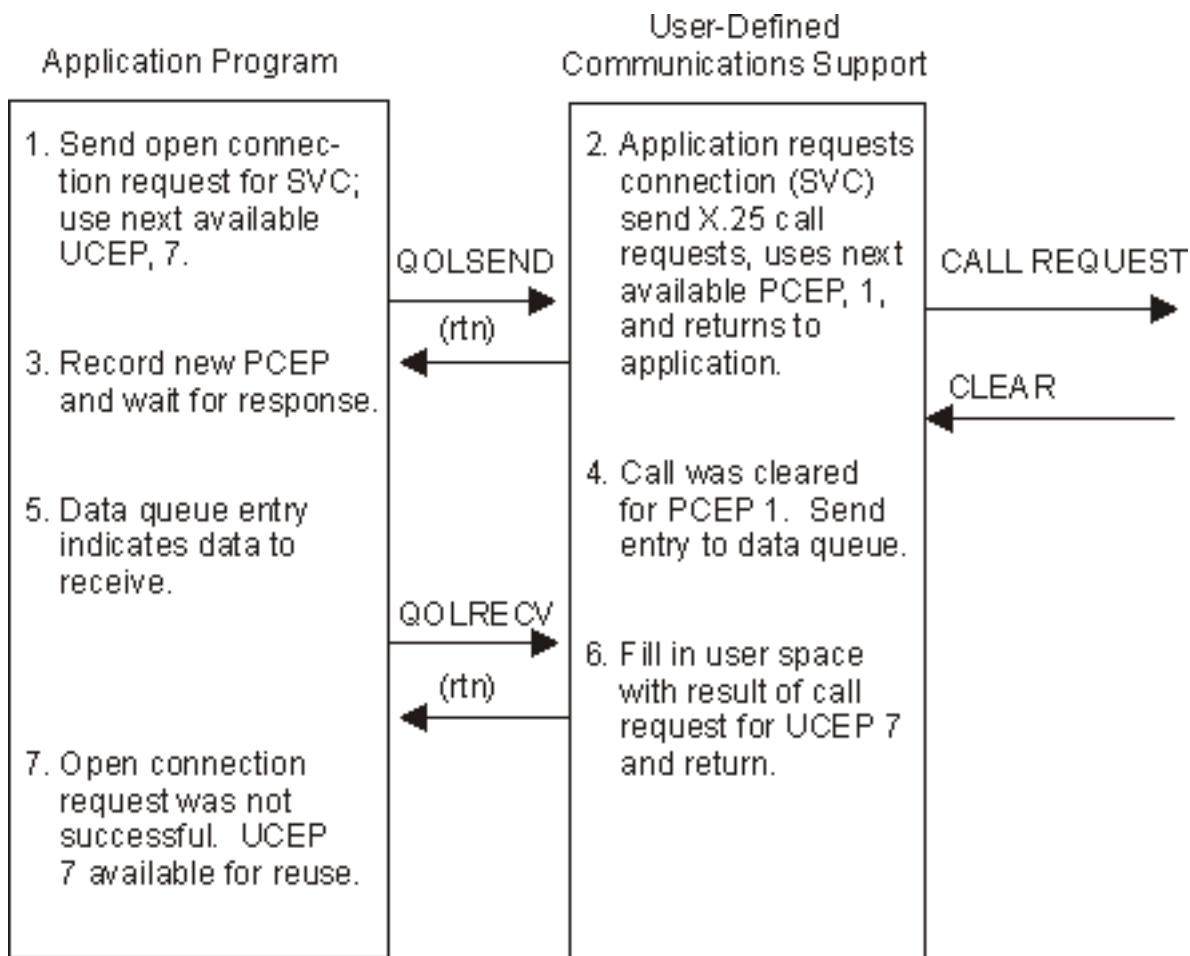
Figure 1-3. Example 1: Normal Connection Establishment



1. The application wants to open a connection, so it calls the qolsend API passing it the UCEP it wants to use for the connection. The application keeps track of the UCEP, PCEP pair. At this point, the UCEP=7, and the PCEP is undefined.

2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP, which is 1, and returns to the application, acknowledging the receipt of the request.  
The user-defined communications support validates the request and issues the X.25 call request.
3. The application records that the PCEP for UCEP=7 is 1. The UCEP=7, PCEP=1 connection is not yet active. Next, the application calls the Receive Entry From Data Queue (QRCVDTAQ) API, to wait for the incoming data entry. The application is expecting the open connection response.
4. The X.25 call accept is received for PCEP=1. To inform the application of the incoming data, an incoming data entry is sent to the data queue.
5. The application's call to the QRCVDTAQ API returns with the incoming data entry. The application then issues a call to the QOLRECV API.
6. The user-defined communications support fills in the input buffer and descriptor with data for the open connection response operation, and determines the UCEP associated with the data by examining the PCEP associated with the X.25 call accept. Because the call accept was received for PCEP=1, the UCEP=7.
7. The application's call to the QOLRECV API returns with successful return and reason codes for the open connection response operation. This operation was reported for UCEP 7; the UCEP=7, PCEP=1 connection is now active.

Figure 1-4. Example 2: Connection Request Cleared by Network/Remote System



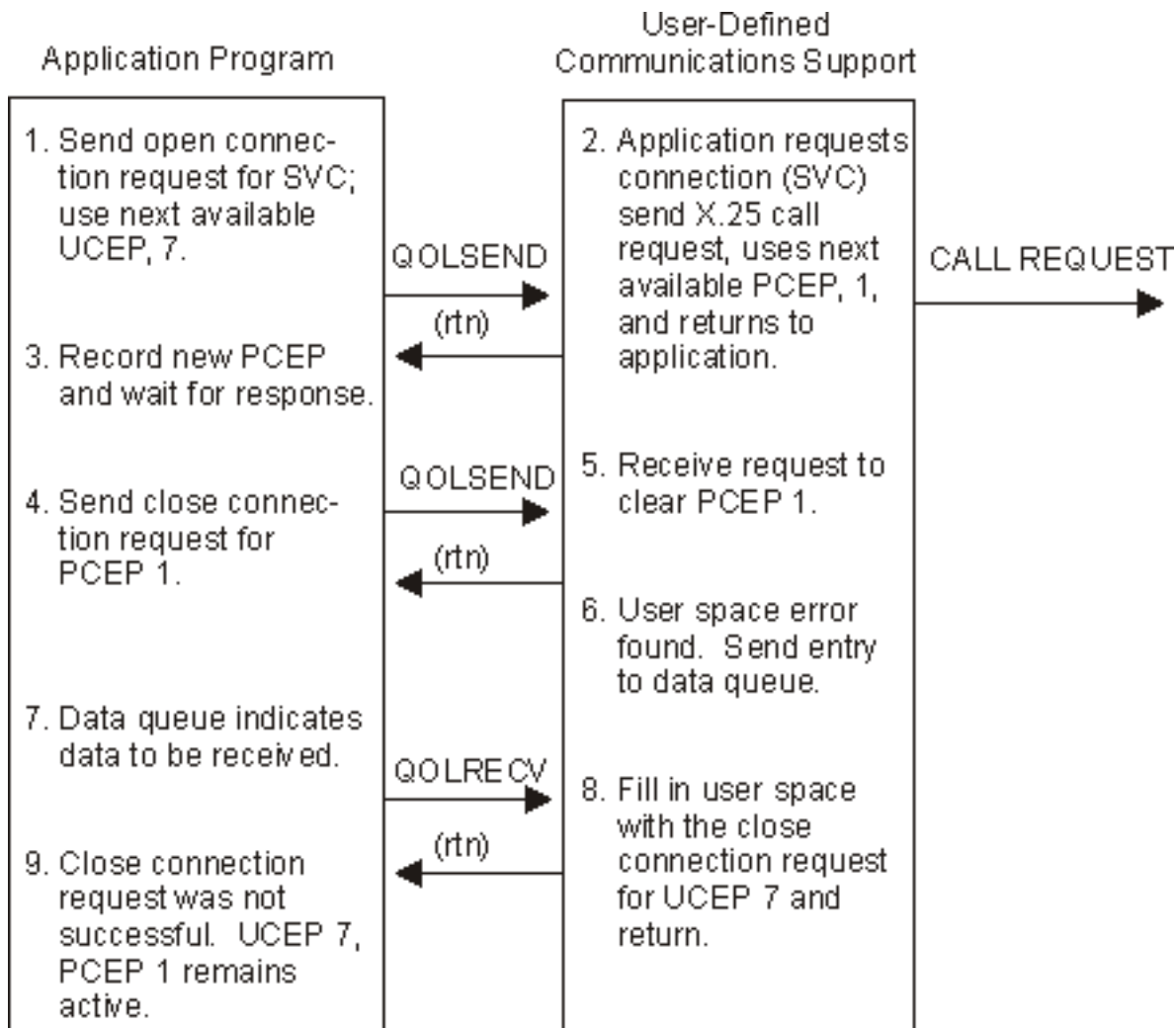
1. The application wishes to open a connection, so it calls the qolsend API, passing it the UCEP it wants to use for the new connection. The application keeps track of the UCEP, PCEP pair. At this point, the UCEP=7, and the PCEP is undefined.

2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP, which is 1, and returns to the application, acknowledging the receipt of the request.

The user-defined communications support validates the request and issues the X.25 call request.

3. The application records that the PCEP for UCEP=7 is 1, and the UCEP=7, PCEP=1 connection is not yet active. Next, the application calls the QRCVDTAQ API to wait for the incoming data entry. The application is expecting the open-connection response.
4. A clear is received for PCEP=1. To inform the application of the incoming data, an incoming data entry is sent to the data queue.
5. The application's call to the QRCVDTAQ API returns with the incoming data entry. The application then issues a call to the QOLRECV API.
6. The user-defined communications support fills in the input buffer and descriptor with data for the open connection response operation, and determines the UCEP for the data by using the PCEP for which the X.25 call accept was received. Because the call was cleared for PCEP=1, the UCEP=7. The PCEP=1 is no longer active, and may be reused by the user-defined communications support.
7. The application's call to the QOLRECV API returns with unsuccessful return and reason codes for the open connection response operation. Thus for the PCEP=1, the UCEP=7. The PCEP=1 is no longer active, and the operation is for UCEP=7. Because the connection is not open, the user-defined communications support's PCEP=1 no longer implies UCEP=7, and the application's UCEP=7 may be reused.

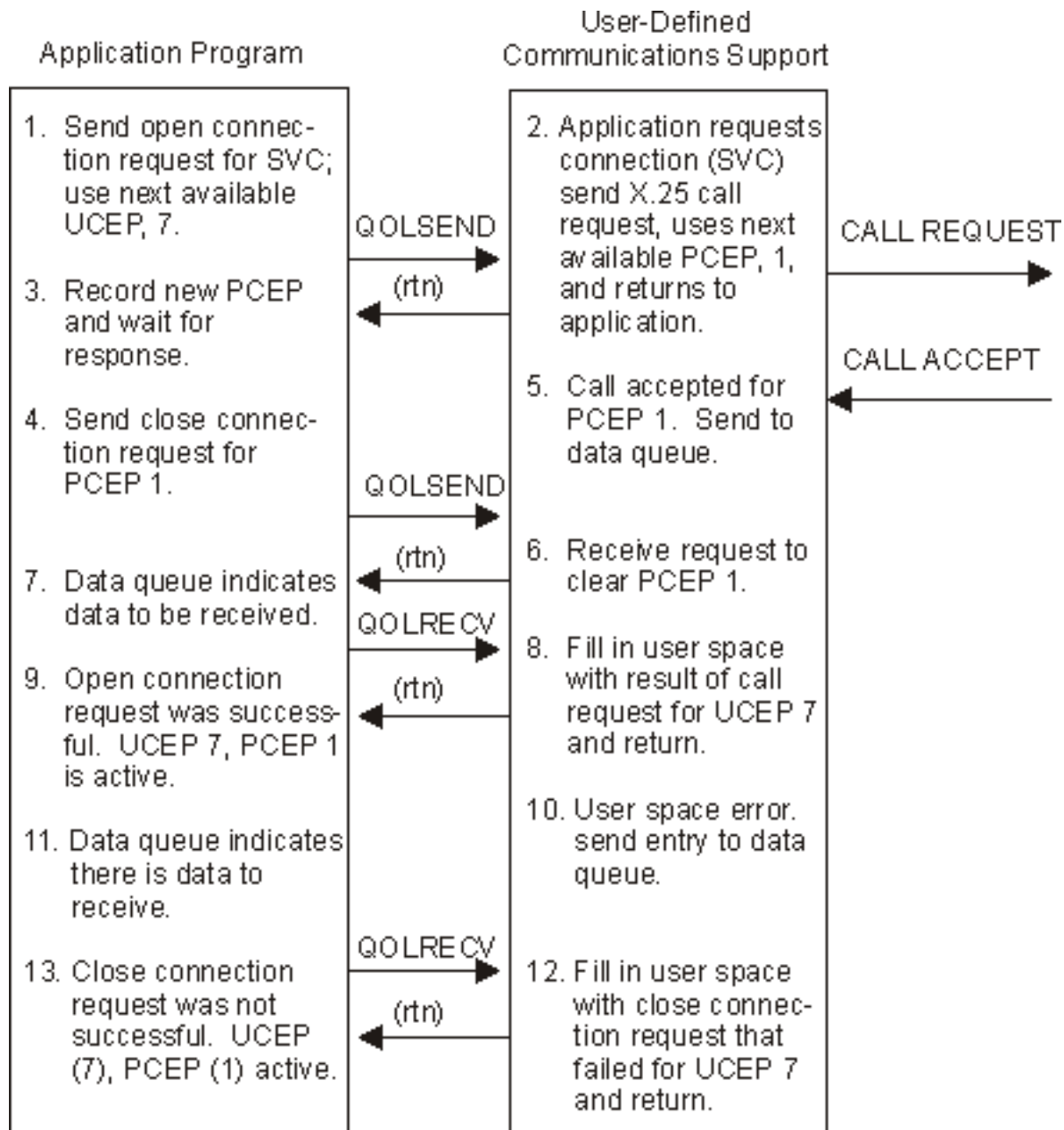
**Figure 1-5. Example 3: Request to Clear Connection with Outstanding Call (Unsuccessful)**



1. The application wishes to open a connection, so it calls the qolsend API passing it the UCEP for the new connection. The application keeps track of the UCEP, PCEP pair. At this point, the UCEP=7, and the PCEP is undefined.
2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP, which is 1, and returns to the application, acknowledging the receipt of the request.  
The user-defined communications support validates the request and issues the X.25 call request.
3. The application records that the PCEP for UCEP=7 is 1, and the UCEP=7, PCEP=1 connection is not yet active. Next, the application calls the QRCVDTAQ API to wait for the incoming data entry. The application is expecting the open connection response.
4. QRCVDTAQ returns to the application (the dequeue time-out value has elapsed), and the application no longer wants the UCEP=7 connection. It calls the qolsend API passing the PCEP=1 to identify the connection to be closed. Then the application calls the QRCVDTAQ API.
5. The user-defined communications support receives the close connection request, and returns to the application, acknowledging the receipt of the request.  
The user-defined communications support validates the request and finds an error.
6. The user space error is found. A copy of the user space, which contained an error, is passed back to the application. To inform the application of the unsuccessful close connection request, an incoming data entry is sent to the data queue.
7. The application's call to the QRCVDTAQ API returns with the incoming data entry. The application then issues a call to the QOLRECV API.

8. The user-defined communications support fills in the input buffer and descriptor with data for the unsuccessful close connection request operation, and determines the UCEP associated with the data by examining the PCEP associated with the close connection. Because the close connection request was for PCEP=1, the UCEP=7.
9. The application's call to the QOLRECV API returns with unsuccessful return and reason codes for the close connection response operation. This operation is for UCEP=7. The connection UCEP=7, PCEP=1 is still in use by both the application and the user-defined communications support. The application can either correct the error and reissue the operation, or wait for the call to be accepted or rejected.

**Figure 1-6. Unsuccessful Attempt to Clear Outstanding (Successful) Call**

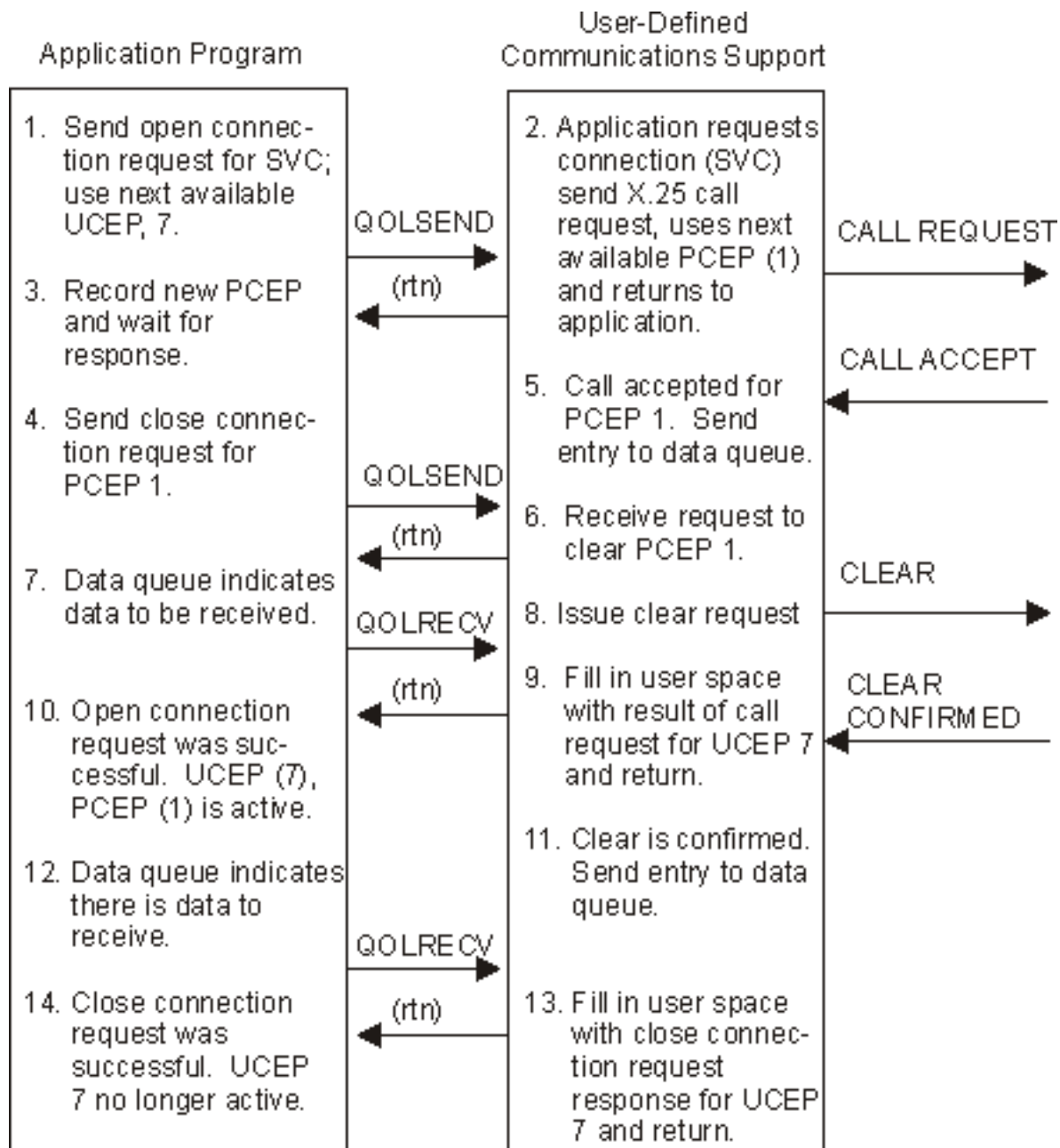


1. The application wishes to open a connection, so it calls the qolSEND API, passing the UCEP for the new connection. The application keeps track of the UCEP, PCEP pair. At this point, the UCEP=7, and the PCEP is undefined.

2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP, which is 1, and returns to the application, acknowledging the receipt of the request.  
The user-defined communications support validates the request and issues the X.25 call request.
3. The application records that the PCEP for UCEP=7 is 1, and the UCEP=7, PCEP=1 connection is not yet active. Next, the application calls the QRCVDTAQ API to wait for the incoming data entry. The application is expecting the open connection response.
4. QRCVDTAQ returns to the application (the dequeue time-out value has elapsed), and the application no longer wants the UCEP=7 connection. It calls the qolsend API passing the PCEP=1 to identify the connection to be closed. Then the application calls the QRCVDTAQ API.
5. The X.25 call accept is received for PCEP=1. To inform the application of the incoming data, an incoming data entry is sent to the data queue.
6. The user-defined communications support receives the close connection request, and returns to the application, acknowledging the receipt of the request.
7. The application's call to the QRCVDTAQ API returns with the incoming data entry. The application then issues a call to the QOLRECV API.
8. The user-defined communications support fills in the input buffer and descriptor with data for the open connection response, and determines the UCEP associated with the data by examining the PCEP for the X.25 call accept. Because the call accept was received for PCEP=1, the UCEP=7.
9. The application's call to the QOLRECV API returns with successful return and reason codes for the open connection request operation. This operation is reported for UCEP=7; the UCEP=7, PCEP=1 connection is now active with an outstanding close connection request. The application calls the QRCVDTAQ API.
10. While processing the close connection request, the user-defined communications support detects an error in the user space. The user space that is in error is copied into the input buffer and descriptor, so the application is aware of the data in error. To inform the application of the unsuccessful close connection request, an incoming data entry is sent to the data queue.
11. The application's call to the QRCVDTAQ API returns with the incoming data entry. The application then issues a call to the QOLRECV API.
12. The user-defined communications support fills the input buffer and descriptor with data for the unsuccessful close connection request operation. By using the PCEP that was requested for the close connection, the support determines the UCEP with which the data is associated. Because the close connection request was for PCEP=1, the UCEP is 7. The PCEP=1 is still active.
13. The application's call to the QOLRECV API returns with unsuccessful return and reason codes for the close connection response operation. This operation is for UCEP 7. The connection UCEP=7, PCEP=1 is still in use by both the application and the user-defined communications support. The application can either correct the error and reissue the operation, or wait for the call to be accepted or rejected.

**Figure 1-7. Example 5: Successful Attempt to Clear Outstanding (Successful) Call**

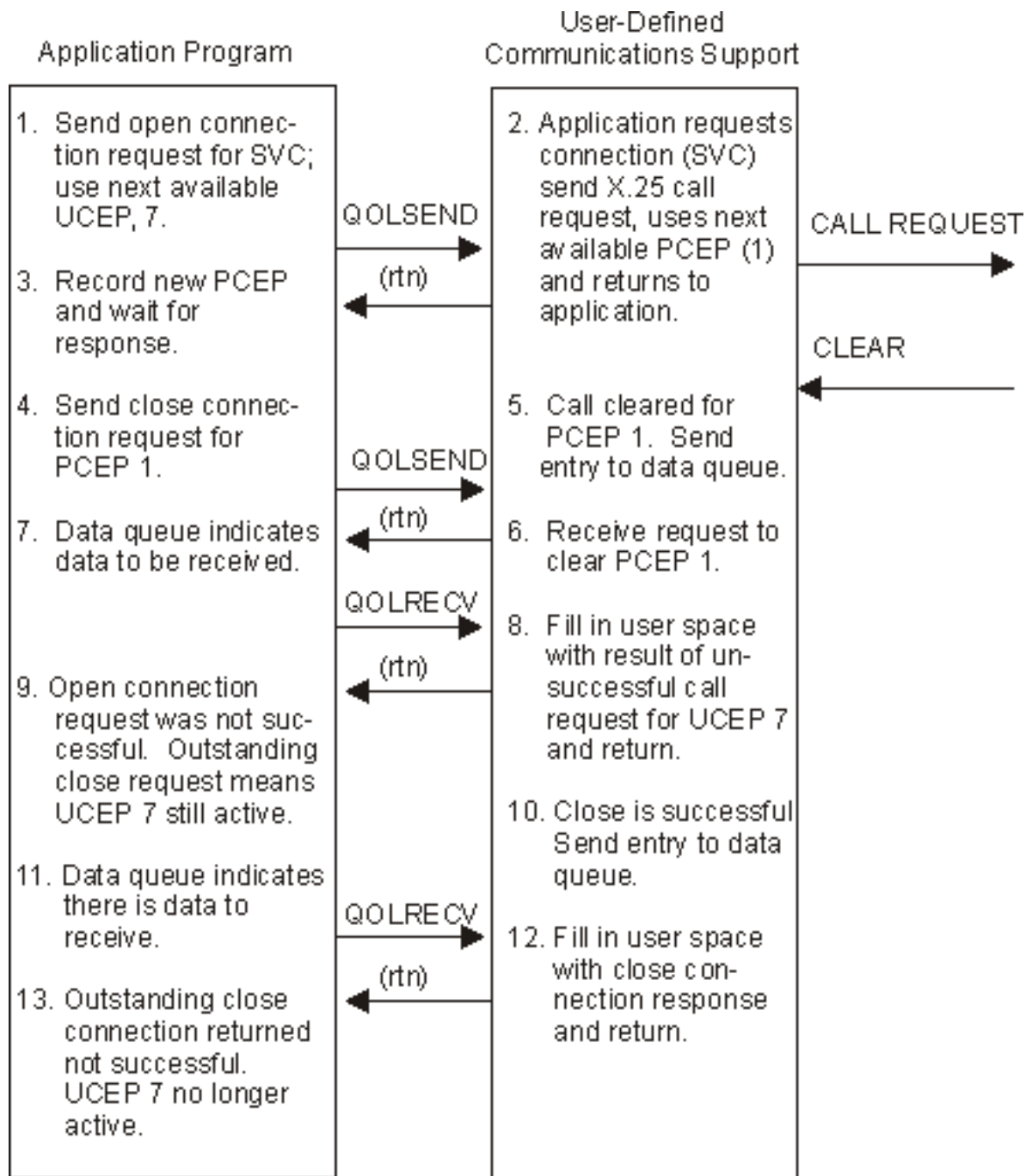




1. The application wishes to open a connection so it calls the qolsend API, passing it the UCEP for the new connection. The application keeps track of the UCEP, PCEP pair. At this point, the UCEP=7, and the PCEP is undefined.
2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP, which is 1, and returns to the application, acknowledging the receipt of the request.  
The user-defined communications support validates the request and issues the X.25 call request.
3. The application records that the PCEP for UCEP=7 is 1, and the UCEP=7, PCEP=1 connection is not yet active. Next, the application calls the QRCVDTAQ API to wait for the incoming data entry. The application is expecting the open connection response.
4. QRCVDTAQ returns to the application (the dequeue time-out value has elapsed), and the application no longer wants the UCEP=7 connection. It calls the qolsend API passing the PCEP=1 to identify the connection to be closed. The application calls the QRCVDTAQ API.

5. The X.25 call-accept is received for PCEP=1. To inform the application of the incoming data, an incoming data entry is sent to the data queue.
6. The user-defined communications support receives the close connection request, and returns to the application, acknowledging the receipt of the request. The application calls QRCVDTAQ API.
7. The application's call to the QRCVDTAQ API returns with the incoming data entry. The application then issues a call to QOLRECV.
8. The user-defined communications support validates the close connection request, and issues an X.25 clear request.
9. The user-defined communications support fills in the input buffer and descriptor with data for the open connection response, and determines the UCEP that the data is for by using the PCEP for the X.25 call accept. Since the call accept was received for PCEP=1, the UCEP is 7.
10. The application's call to QOLRECV returns with successful return and reason codes for the open connection request operation. This operation is reported for UCEP=7; the UCEP=7, PCEP=1 connection is now active with an outstanding close connection request.
11. The clear confirmation is received for PCEP=1. To inform the application of the successful close connection request, an incoming data entry is sent to the data queue.
12. The application's call to the QRCVDTAQ API returns indicating there is data to receive.
13. The user-defined communications support fills the input buffer and descriptor with data for the successful close connection request operation, and determines the UCEP associated with the data by examining the PCEP that was requested for the close connection. Because the close connection request was for PCEP=1, the UCEP=7. The PCEP=1 is no longer active.
14. The application's call to the QOLRECV API returns with successful return and reason codes for the close connection response operation. This operation is for UCEP 7. The UCEP=7, PCEP=1 connection is no longer active.

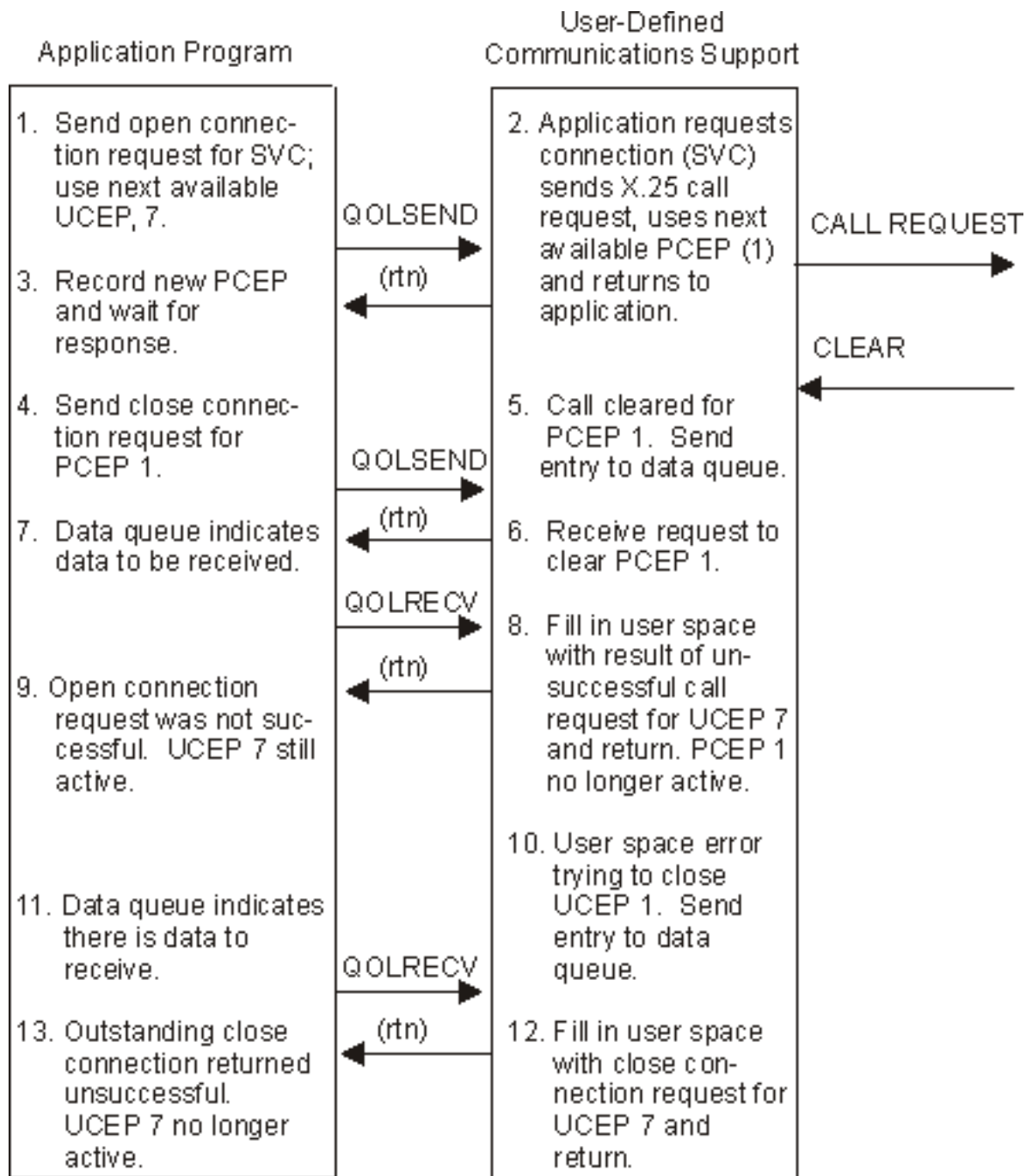
**Figure 1-8. Example 6: Successful Attempt to Clear Outstanding (Unsuccessful) Call**



1. The application wishes to open a connection, so it calls the qolsend API, passing it the UCEP for the new connection. The application keeps track of the UCEP, PCEP pair. At this point, the UCEP=7, and the PCEP is undefined.
2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP, which is 1, and returns to the application, acknowledging the receipt of the request.  
The user-defined communications support validates the request and issues the X.25 call request.
3. The application records that the PCEP for UCEP=7 is 1, and the UCEP=7, PCEP=1 connection is not yet active. Next, the application calls the QRCVDTAQ API to wait for the incoming data entry. The application is expecting the open connection response.

4. QRCVDTAQ API returns to the application (the dequeue time-out value has elapsed), and the application no longer wants the UCEP=7 connection. It calls the qolsend API passing the PCEP=1 to identify the connection to be closed. Then the application calls the QRCVDTAQ API.
5. The X.25 clear is received for PCEP=1. To inform the application of the incoming data, an incoming data entry is sent to the data queue.
6. The user-defined communications support receives the close connection request, and returns to the application, acknowledging the receipt of the request.
7. The application's call to the QRCVDTAQ API returns with the incoming data entry. The application then issues a call to the QOLRECV API.
8. The user-defined communications support fills in the input buffer and descriptor with data for the open connection response, and determines the UCEP that the data is for by using the PCEP that the X.25 clear is for. Because the clear was received for PCEP=1, the UCEP is 7.
9. The application's call to the QOLRECV API returns with unsuccessful return and reason codes for the open connection request operation. This operation is reported for UCEP=7. Because the close connection request is outstanding, the UCEP=7, PCEP=1 connection is not fully closed. The application calls the QRCVDTAQ API.
10. The close connection request is validated, but no clear is sent because the connection was cleared previously. The close is considered successful, and an entry is sent to the data queue.
11. The application's call to the QRCVDTAQ API returns with the incoming data entry. The application then issues a call to the QOLRECV API.
12. The user-defined communications support fills in the input buffer and descriptor with data for the successful close connection request operation, and determines the UCEP that the data is for by using the PCEP that the close connection was requested for. Since the close connection request was for PCEP=1, and the UCEP is 7. The PCEP=1 is no longer active.
13. The application's call to the QOLRECV API returns with unsuccessful return and reason codes for the close connection response operation. This operation is for UCEP 7. The connection UCEP=7, PCEP=1 is no longer active.

**Figure 1-9. Example 7: Unsuccessful Attempt to Clear Outstanding (Unsuccessful) Call**



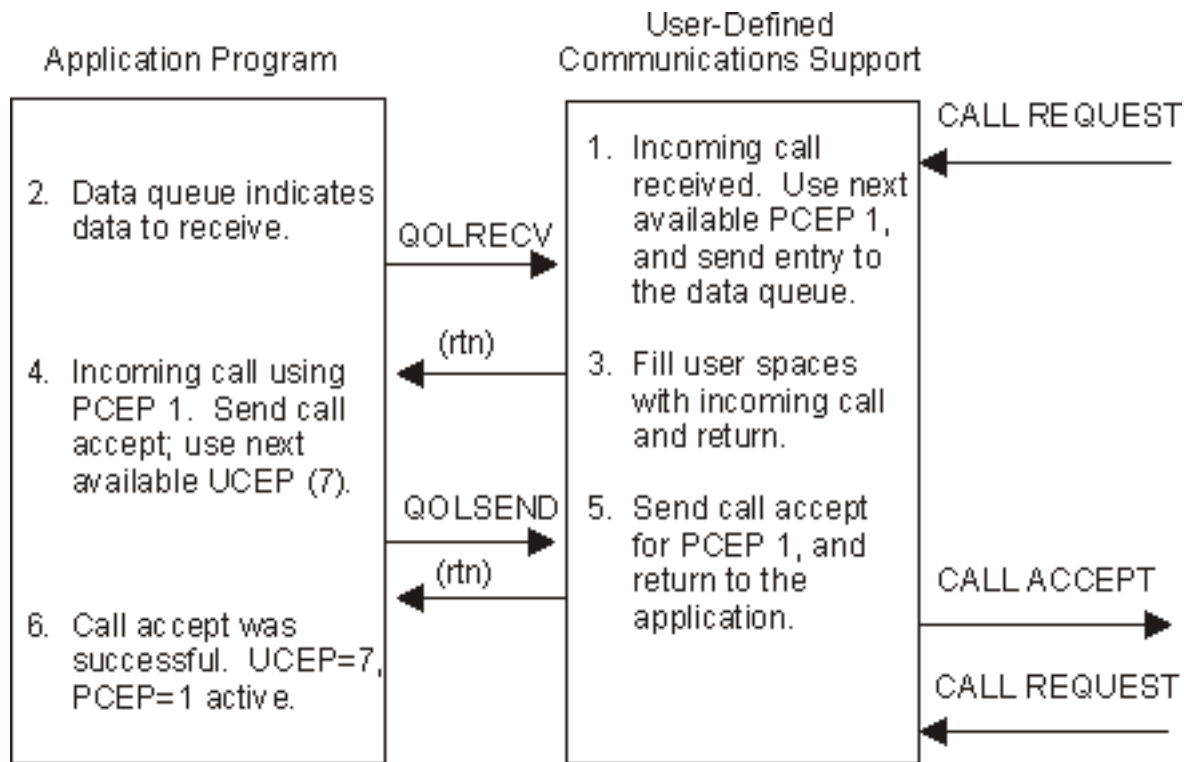
1. The application wishes to open a connection, so it calls the qolsend API passing it the UCEP for the new connection. The application keeps track of the UCEP, PCEP pair. At this point, the UCEP=7, and the PCEP is undefined.
2. The user-defined communications support receives the request, stores the UCEP for the connection, and uses the next available PCEP (1); and returns to the application, acknowledging the receipt of the request.  
The user-defined communications support validates the request and issues the X.25 call request.
3. The application records that the PCEP for UCEP=7 is 1, and the UCEP=7, PCEP=1 connection is not yet active. Next, the application calls the QRCVDTAQ API to wait for the incoming data entry. The application is expecting the open connection response.
4. The application no longer wants the UCEP=7 connection. It calls the qolsend API passing the PCEP=1 to identify the connection to be closed. The application calls the QRCVDTAQ API.

5. The X.25 Clear is received for PCEP=1. To inform the application of the incoming data, an incoming data entry is sent to the data queue.
6. The user-defined communications support receives the close connection request, and returns to the application, acknowledging the receipt of the request.
7. The application's call to the QRCVDTAQ API returns with the incoming data entry. The application then issues a call to the QOLRECV API.
8. The user-defined communications support fills in the input buffer and descriptor with data for the open connection response, and determines the UCEP that the data is for by using the PCEP that the X.25 clear is for. Because the clear was received for PCEP=1, the UCEP is 7.
9. The application's call to the QOLRECV API returns with unsuccessful return and reason codes for the open connection request operation. This operation is reported for UCEP=7. Because the close connection request is outstanding, the UCEP=7, PCEP=1 connection is not fully closed. The application calls the QRCVDTAQ API.
10. The close connection request is validated, and an error is found in the user space. An entry is sent to the data queue.
11. The application's call to the QRCVDTAQ API returns with the incoming data entry. The application then issues a call to the QOLRECV API.
12. The user-defined communications support fills in the input buffer and descriptor with data for the unsuccessful close connection request operation, and determines the UCEP that the data is for by using the PCEP that the close connection was requested for. Since the close connection request was for PCEP=1, the UCEP is 7. Because the connection was cleared prior to the close connection request, the PCEP=1, UCEP=7 connection is considered no longer active to the user-defined communications support.
13. The application's call to the QOLRECV API returns with unsuccessful return and reason codes for the close connection response operation. This operation is for UCEP 7. The connection UCEP=7, PCEP=1 is no longer active.

## Incoming Connections

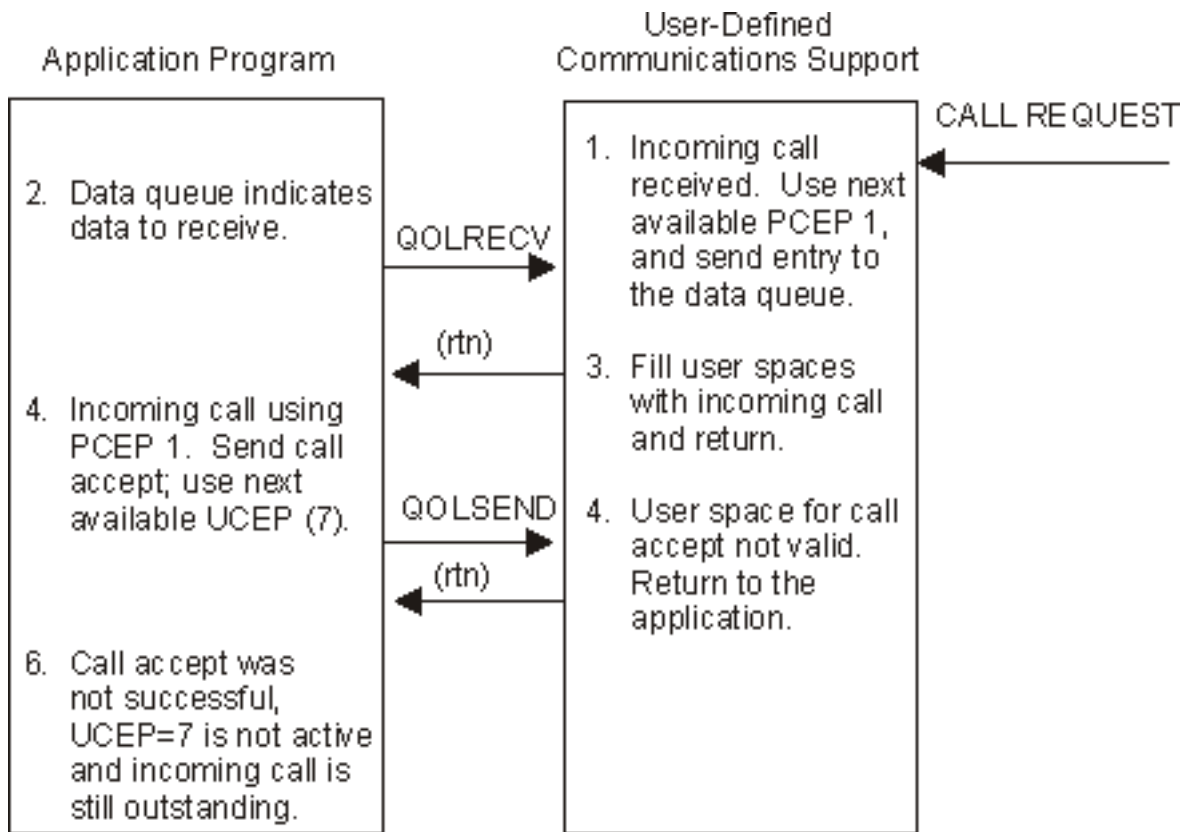
The following figures show how the application program handles UCEPs and PCEPs for incoming connections.

**Figure 1-10. Example 1: Normal Connection Establishment**



1. An incoming call request is received by the communications support, which determines if there is an application that has a filter satisfying this call request. The communications support uses the next available PCEP, which is 1, for this new connection. An entry is sent to the data queue.
2. The application has been waiting for its call to the QRCVDTAQ API to complete. The call completes indicating there is data to be received. The application calls the QOLRECV API.
3. The input buffer and descriptor are filled with the incoming call request for PCEP=1, and the QOLRECV API returns.
4. The application looks at the operation, which indicates an incoming call indication. The PCEP reported by the communications support is 1. The application chooses to accept this call, and passes the UCEP to be used for this new connection. The call is made to the qolsend API with PCEP=1, UCEP=7.
5. The call accept is received and sent for PCEP=1. The qolsend API returns to the application.
6. The call accept request was successful for UCEP=7, PCEP=1. This connection is now active.

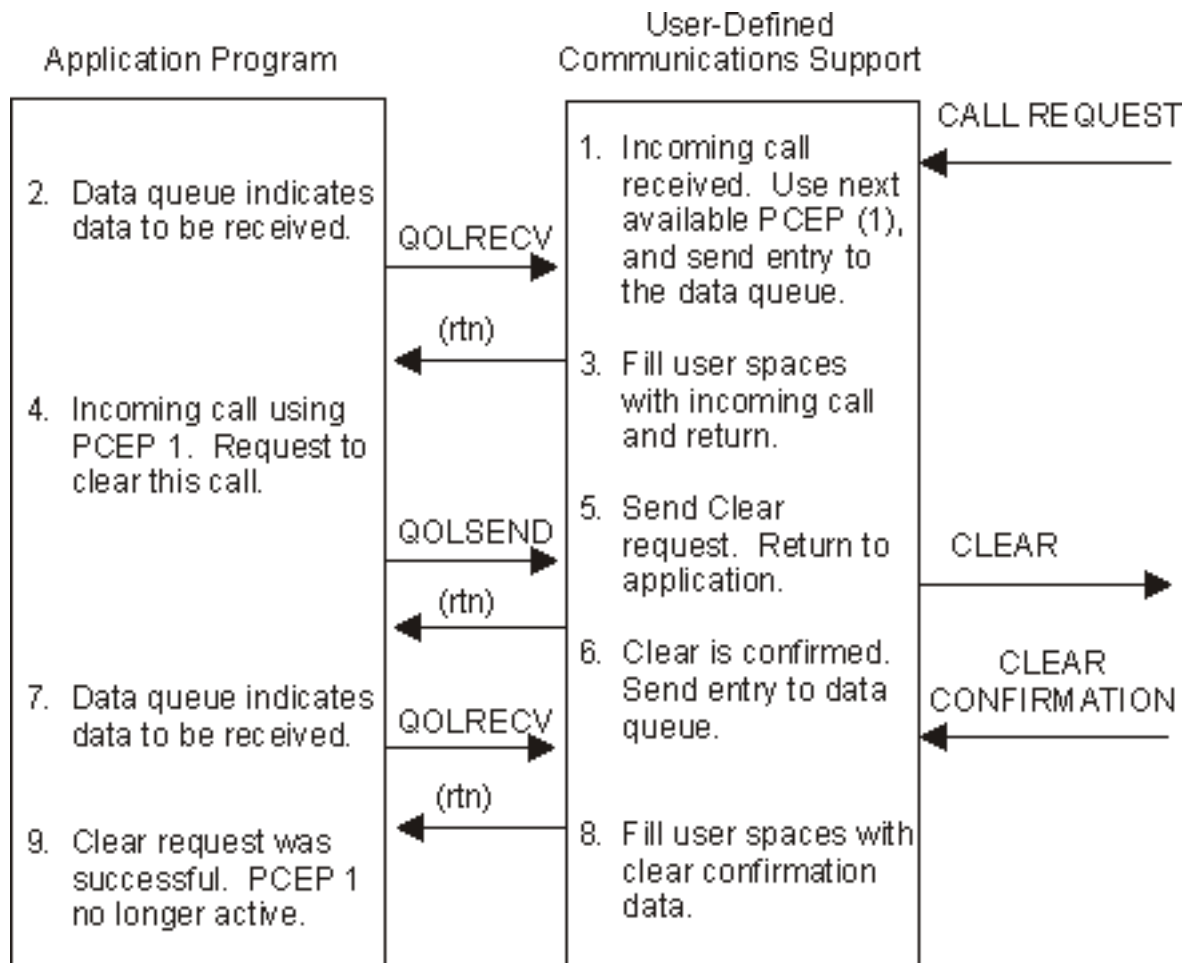
**Figure 1-11. Example 2: Send Call Accept Not Valid**



1. An incoming call request is received by the communications support, which determines if there is an application that has a filter satisfying this call request. The communications support uses the next available PCEP=1 for this new connection. An entry is sent to the data queue.
2. The application has been waiting for its call to the QRCVDTAQ API to complete. It does, indicating there is data to be received. The application calls the QOLRECV API.
3. The input buffer and descriptor are filled with the incoming call request for PCEP=1, and the QOLRECV API returns.
4. The application looks at the operation which indicates an incoming call indication. The PCEP reported by the communications support is 1. The application chooses to accept this call, and passes the UCEP to be used for this new connection. The call is made to the qolsend API with PCEP=1, UCEP=7.
5. The call accept is received and an error is found in the user space. The qolsend API returns to the application, reporting the error and offset. The incoming call is still outstanding for PCEP=1. The application checks the return and reason codes and finds that an error has occurred. The call accept was not sent and the incoming call is still waiting for a response.

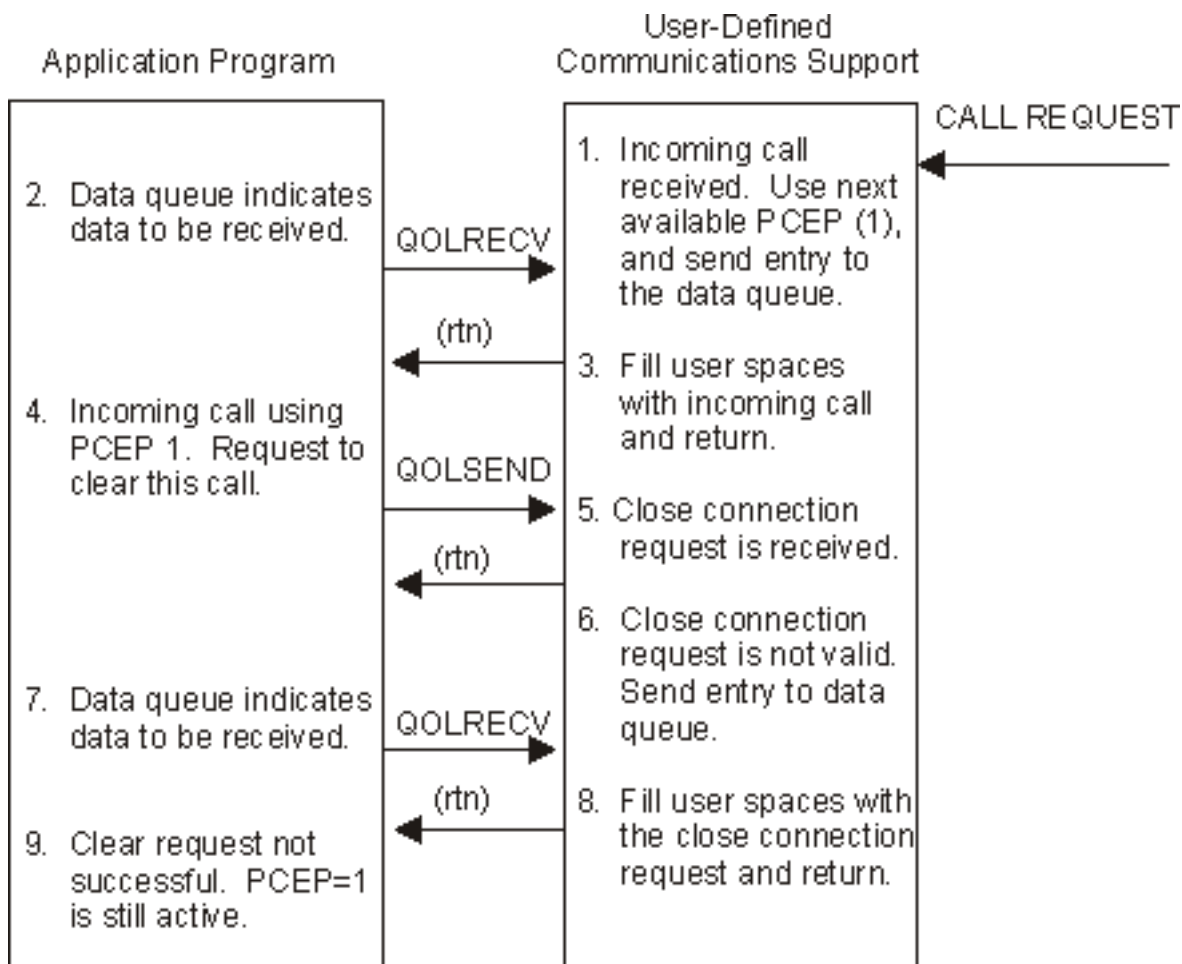
**Figure 1-12. Example 3: Send Clear for Incoming Call**





1. An incoming call request is received by the communications support, which determines if there is an application that has a filter satisfying this call request. The communications support uses the next available PCEP, which is 1, for this new connection. An entry is sent to the data queue.
2. The application has been waiting for its call to the QRCVDTAQ API to complete. It does, indicating there is data to be received. The application calls the QOLRECV API.
3. The input buffer and descriptor are filled for the incoming call request for PCEP=1, and the QOLRECV API returns.
4. The application looks at the operation which indicates an incoming call indication. The PCEP reported by the communications support is 1. The application does not wish to accept the call, so the user space is filled in for a close connection request and the application calls the qolsend API. The application calls the QRCVDTAQ API.
5. The close connection request is received and the qolsend API returns to the application, acknowledging the request.  
The close connection request is validated and a clear is sent.
6. The clear confirmation is received for PCEP=1 which has no UCEP. An incoming data entry is sent to the data queue. The application calls the QRCVDTAQ API.
7. The application's call to the QRCVDTAQ API returns with the incoming data entry. The application calls the QOLRECV API to receive the data.
8. The input buffer and descriptor are filled in with the clear confirmation data. Since the connection was never established (and the application never assigned a UCEP to this connection), the QOLRECV API returns to the application passing a UCEP=0.
9. The close connection request was successful. PCEP=1 is no longer active.

Figure 1-13. Example 4: Send Clear for Incoming Call



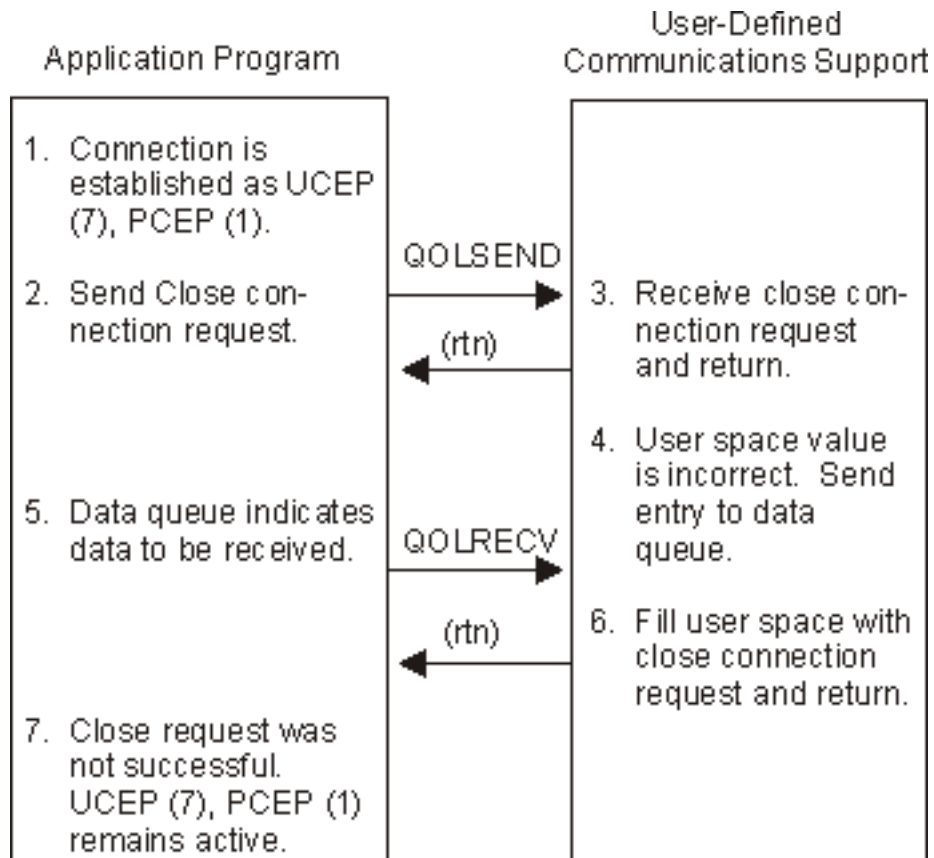
1. An incoming call request is received by the communications support, which determines there is an application that has a filter satisfying this call request. The communications support uses the next available PCEP=1 for this new connection. An entry is sent to the data queue.
2. The application has been waiting for its call to the QRCVDTAQ API to complete. It completes indicating there is data to be received. The application calls the QOLRECV API.
3. The input buffer and descriptor are filled for the incoming call request for PCEP=1, and the QOLRECV API returns.
4. The application looks at the operation which indicates an incoming call indication. The PCEP reported by the communications support is 1. The application does not wish to accept the call, so the user space is filled in for a close connection request and the qolsend API. The application calls the QRCVDTAQ API.
5. The close connection request is received and the qolsend API returns to the application, acknowledging the request.
6. The close connection request is validated and an error is found. An entry is sent to the data queue.
7. The application's call to the QRCVDTAQ API return, with the incoming data entry. The application calls the QOLRECV API to receive the data.
8. The input buffer and descriptor are filled in with the unsuccessful close request, and the QOLRECV API returns to the application.
9. The close connection request was not successful. PCEP=1 is still active.

## Closing Connections

The following figures show how the application program closes a connection. The figures apply to both incoming and outgoing connections.

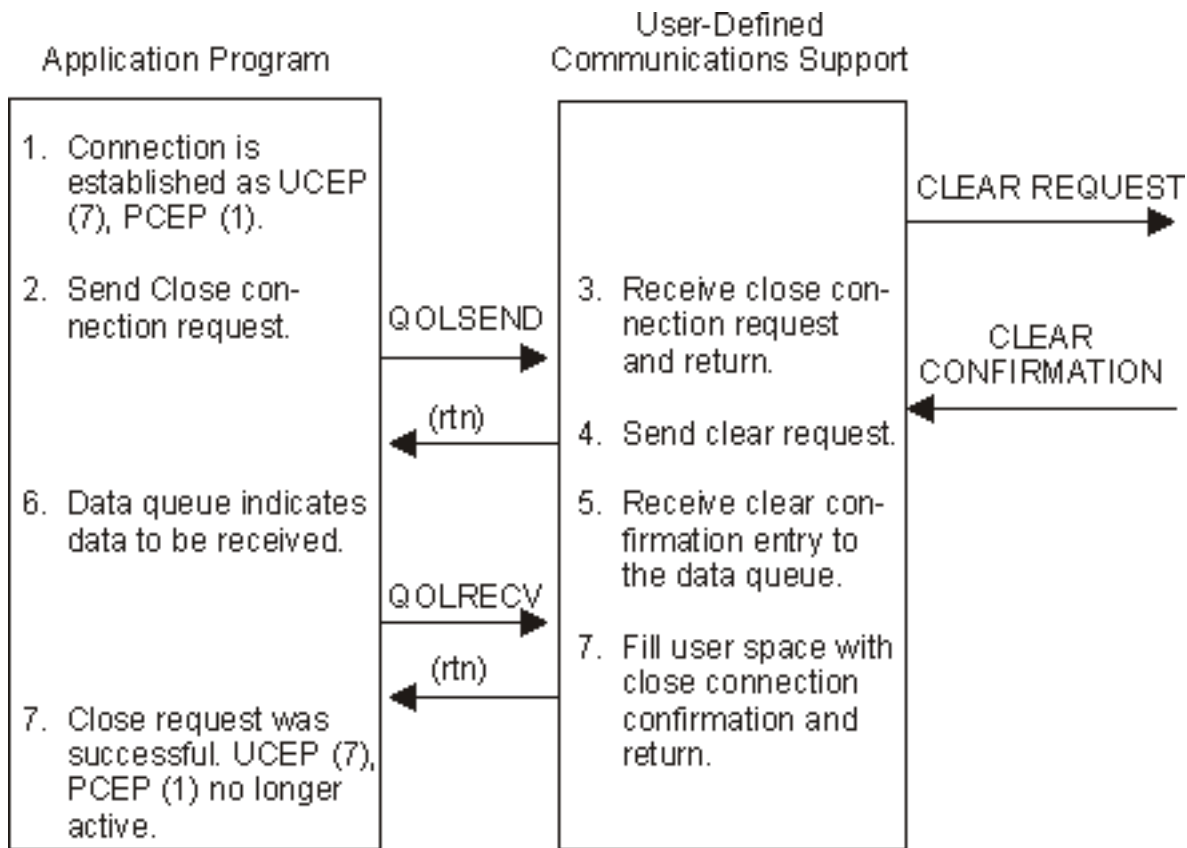
The next two figures illustrate that a close connection request never completely guarantees the connection will be closed.

Figure 1-14. Example 1: Close Connection Request Is Not Valid



1. A connection is established with the PCEP=1, UCEP=7.
2. The application calls the `qolsend` API to close the connection. The application calls the `QRCVDTAQ` API.
3. The user-defined communications support receives the close connection request and returns to the application, acknowledging the receipt of the request.
4. The value in the user space is not correct. An entry is sent to the data queue.
5. The application's call to the `QRCVDTAQ` API returns with the incoming data entry. The application calls the `QOLRECV` API to receive the data.
6. The user-defined communications support fills the input user space with data for the close connection request and determines the UCEP that the data is for by examining the PCEP that was requested for the close connection.
7. The application's call to the `QOLRECV` API returns with unsuccessful return and reason codes for the close connection response. This operation is for UCEP 7. The connection UCEP=7, PCEP=1 is still active.

Figure 1-15. Example 2: Close Connection Request Is Valid




1. A connection is established with the PCEP=1, UCEP=7.
2. The application calls the qolsend API to close the connection. The application calls the QRCVDTAQ API.
3. The user-defined communications support receives the close connection request and returns to the application, acknowledging the receipt of the request.
4. The close connection request is received and the qolsend API returns to the application, acknowledging the request. The close connection request is validated and a clear is sent.
5. The clear confirmation is received for PCEP=1, UCEP=7. An incoming data entry is sent to the data queue.
6. The application's call to the QRCVDTAQ API returns with the incoming data entry. The application calls the QOLRECV API to receive the data.
7. The user-defined communications support fills the input user space with data for the close connection confirmation and determines the UCEP that the data is for by examining the PCEP that was requested for the close connection.
8. The application's call to the QOLRECV API returns with successful return and reason codes for the close connection response. This operation is for UCEP 7. The connection UCEP=7, PCEP=1 is no longer active.

---

## Programming Considerations for LAN Applications

User-defined communications over LANs use connectionless (unacknowledged) service. Unacknowledged Information (UI) frames are the only frames an application program can generate.

For a description of the frame formats for Ethernet Version 2, IEEE 802.3, IEEE 802.5, wireless, and FDDI, refer to the LAN, Frame-Relay and ATM Support  book. To determine how the format and the user buffer are specified, see "User-Defined Communications Support APIs" on page 1.

## Operations

User-defined communications support defines many different operations. Not all operations are valid on all data links. The operations which are valid for LAN links are:

- X'0000' and X'0001'. These operations together represent the send- and receive-data operations for any of the LAN frames types.

## Configuration

The service access point (SAP) that the application program uses to send and receive data must be configured in the line description. The 04, 06, and AA SAPs are created if \*SYSGEN is specified on the CRTLINTRN, CRTLINETH, CRTLINWLS, or CRTLINDDI command. The 04 SAP is used by SNA, and the 06 and AA SAPs are used by TCP/IP. An application can choose to use any SAP (including SAPs defined by SNA or IEEE). The line description must be manually configured to include any other SAPs the application uses. The SAPTYPE for each SAP used must be configured as \*NONSNA to be used by user-defined communications.

Although it is possible to use any SAP configurable on the iSeries server, it is not recommended to use SNA SAPs for user-defined communications, because this may restrict the use of SNA on your iSeries server. In the same manner, using the same SAP as other well-known protocols, such as TCP/IP, may restrict the use of these protocols or the application program on the iSeries server.

**Note:** It is not possible to run an SNA application and a user-defined communications application program over the same SAP concurrently. It is possible to run a TCP/IP application and a user-defined communications application over the same SAP concurrently, provided the inbound routing information is unique among all the non-SNA applications sharing the network controller.

## Inbound Routing Information

For an application program to receive data from a LAN, it must inform the communications support of how to filter the inbound data and route it to the application. This is accomplished by a program call to the Set Filter (QOLSETF) API. The fields in the incoming frame that are used to route the data are DSAP, SSAP, MAC address, and type.

The inbound routing information acts as a filter to allow the user-defined application to distinguish its data from the rest of the data on the LAN. The more selective the inbound routing information is, the less chance there is that the application will be processing unnecessary input requests. Also, more selective inbound routing information allows multiple jobs running user-defined communications applications to share the same SAP.

For example, if an application is using 92 SSAP and 92 DSAP but only talking to one remote system, it may want to set a more selective filter which would include DSAP, SSAP, and the MAC address of the remote system. Conversely, if an application accepts data on the 04 SAP from all systems sending data on any SAP, then the application would set a filter for DSAP only, indicating that it will accept all data arriving on the 04 SAP.

## End-to-End Connectivity

Because user-defined communications on a LAN is connectionless, it is up to your application protocol to define a method to reach the remote systems it communicates with. There are several ways to do this. One way is to have each system configured in a database file on the iSeries server. Each system could have a local name that the application program uses to correlate with the MAC address and routing information. LANs provide a technique to broadcast, which can be used to retrieve this information as well. An example of this is the Address Resolution Protocol (ARP) used by TCP/IP, which returns the MAC address and routing information so that a system without that information can communicate with a new remote system.

## Sending and Receiving Data

### Maximum Frame Size

The user-defined communications support creates a data unit size which is always large enough to contain the maximum frame size supported by any of the SAPs configured for non-SNA use, (SAPTYPE(\*NONSNA)). The data unit size is returned in the parameter list on the call to the QOLELINK API. For Ethernet (802.3), token-ring, FDDI, and wireless LANs, the maximum frame size that the application can specify is the maximum frame size allowed by the SAP that the frame is sent on. There is no minimum frame size for the Ethernet 802.3, token-ring, FDDI, or wireless LANs.

Ethernet Version 2 does not define SAPs for the higher-layer protocols. Therefore, the maximum frame size is not determined by the maximum frame size for the SAP that the frame is sent on. The maximum frame size for Ethernet Version 2 is 1502 bytes. The first 2 bytes are for the type field, and the last 1500 bytes are for user data. The minimum amount of data that can be sent is 48 bytes. The first 2 bytes are for the type field, and the next 46 bytes are for user data. If the line is configured to handle both Ethernet 802.3 and Ethernet Version 2 data, the larger of the configured value or 1502 bytes is chosen and reported to the application on the data unit size parameter returned from the QOLELINK API.

If your application program attempts to send data frames that are larger or smaller than those that are supported, the output request completes with nonzero return and reason codes, and an error code is returned to the application in the diagnostic information field.

Application programs access information that is contained in the line description through the Query Line Description (QOLQLIND) API. It is best to call the QOLQLIND API after the link has been successfully enabled because the information that the QOLQLIND API passes to the application is accurate for as long as the link is enabled. The application uses the information on the frame size for the SAP to send the correct amount of data over the SAP.

### Maximum Amount of Outstanding Data

Most often, the data arrives at a slightly faster rate than the application program can receive it. The communications support keeps data intended for an application so that the application can receive it. However, there is a limit to the amount of data that can be kept for the application to use later. The limit helps to avoid one system from overrunning another system's resources. When this limit is reached, all new incoming data frames for that application are discarded until the application picks up one third of the data that was stored for the application. Because the data consists of unacknowledged information frames, the higher-layer protocol within the application detects the loss of data, resends the data, or performs other recovery actions.

Each time the data limit is exceeded, the communications support creates an error log entry and puts a message in the QSYSOPR message queue, indicating that the unacknowledged service has temporarily stopped receiving incoming frames.

### Ethernet to Token-Ring Conversion and Routing

The IBM<sup>(R)</sup> 8209 Ethernet to token-ring bridge provides additional connectivity options for the iSeries server. See the *IBM 8209 LAN Bridge Customer Information* book, SA21-9994, for more details.

### Performance Considerations

The application program enables connectionless traffic to enter the iSeries server system from the LAN. In the call to the QOLSETF API, the DSAP field indicates the SAP which will be activated on the iSeries server. By activating traffic over a SAP, data is taken from the LAN and brought into the iSeries server. Similarly, deactivating traffic over an SAP causes traffic intended for that SAP to be left at the IOP level rather than to be processed on the iSeries server system.

To minimize host processing, the SAP or SAPs that the application uses should be deactivated as soon as the application no longer wants to receive traffic for the SAP. If the link is disabled and no other applications are using the SAP(s), they are deactivated automatically by the user-defined communications support.

Protocols that use broadcast frames as a discovery technique could flood the network with messages and affect performance on all the systems attached to the network.

## Programming Considerations for X.25 Applications

The user-defined communications support interface to an X.25 network is at the packet level, which is a connection-oriented level. Your application program is responsible for ensuring reliable end-to-end connectivity. **End-to-end connectivity** means that the application program can initiate, receive, and accept X.25 calls and handle network errors reported to the application, as well as send and receive data.

Your application program has access to packets that flow over switched virtual circuits (SVCs) and permanent virtual circuits (PVCs). The application can have SVC and PVC connections active concurrently. You can configure up to 64 virtual circuits on an X.25 line description, depending on the

communications I/O processor used. The X.25 Network Support  book provides more information about configuration limitations.

The Display Connection Status (DSPCNNSTS) command shows the virtual circuits that are in use by a network device, and the state of each connection. This command also displays the active inbound routing information that the application program uses to route calls.

## X.25 Packet Types Supported

A **packet** is the basic unit of information transmitted through an X.25 network. The following table lists the X.25 packet types along with the type of service provided. Services for Switched Virtual Circuit (SVC) and Permanent Virtual Circuit (PVC) connections are identified as well as services that are not accessible (N/A) to an application program.

Packet Type	Application Input or Access	SVC	PVC	N/A
Data	Q,D bits of the general format identifier (GFI)  <b>Note:</b> The modulus used is configured in the line description. The open connection request allows the user-defined communications support to set the actual window size used.	X	X	
Interrupt	32 bytes of data  <b>Note:</b> On the iSeries server, the X.25 packet layer provides the confirmation of the receipt of this packet. The call to the qolsend API does not return until the interrupt is confirmed by the remote system.	X	X	
Reset request	Cause and diagnostic codes  <b>Note:</b> The application program provides the confirmation of this packet.	X	X	
Reset indication	Cause and diagnostic codes  <b>Note:</b> The application program provides the confirmation of this packet.	X	X	
Reset confirmation	<b>Note:</b> User-defined communications support detects and reports reset collisions to the application on the reset confirmation.	X	X	
Incoming Call	Remote DTE, local virtual circuit, packet and window sizes, up to 109 bytes of additional facilities, up to 128 bytes of bytes of call user data	X		
Call Request	Remote DTE, local virtual circuit, packet and window sizes, up to 109 bytes of additional facilities, up to 128 bytes of bytes of call user data	X		
Call Accept	Packet and window sizes, up to 109 bytes of additional facilities	X		
Call Connected	Negotiated packet and window sizes, facilities	X		

Packet Type	Application Input or Access	SVC	PVC	N/A
Clear request	Cause and diagnostic codes, facilities, up to 128 bytes of clear user data	X		
Clear indication	Cause and diagnostic codes, facilities, up to 128 bytes of clear user data  <b>Note:</b> The X.25 packet layer support provides the confirmation on this request.	X		
Clear confirmation	The X.25 packet layer support provides this support.	X		
Receive Ready (RR)	The flow of RR and RNR packets is determined by the automatic flow control field of Format I, specified in the open connection request.			X
Receive Not Ready (RNR)	The flow of RR and RNR packets is determined by the automatic flow control field of Format I, specified in the open connection request.			X
Reject (REJ)	This packet is not necessarily available on all networks and is not supported by the iSeries server.			X
Restart Request, Indication, and Confirmation	These packets affect all virtual circuits on the line.			X
Diagnostic	This packet is not necessarily available on all networks and is not supported by the iSeries server.			X
Registration Request and Confirmation	This packet is not necessarily available on all networks and is not supported by the iSeries server.			X

## Operations

User-defined communications support defines many different operations. The X'B000' operation either initiates an X.25 SVC call request, or a request to open a PVC. By using this operation, an application program initiates an open connection request. The X'B100' operation either initiates an X.25 SVC clear request (or confirms the connection failure), or requests closing a PVC. By using this operation, an application program initiates a close connection request. The application can use the X'BF00' operation to cause the SVC or PVC connection to be reset.

The open connection request, close connection request, and reset request (or response) operations are two-step operations. See "Synchronous and Asynchronous Operations" on page 250 for more information on programming for two-step operations.

The X'B400' operation initiates an X.25 SVC call accept. This operation is known as a call accept operation. The X'0000' operation initiates an X.25 Data packet for a SVC or PVC connection. This operation is called a send data operation. The call accept and send data operations are one-step operations. See "Synchronous and Asynchronous Operations" on page 250 for more information on programming for one step operations.

The application program does not request the other available X.25 operations. These X.25 operations are inbound packets for responses from the asynchronous operations that are reported to the application in the parameter list of the QOLRECV API. The X'B201' operation indicates an incoming X.25 SVC call and is known as the call indication operation. The X'B301' operation indicates that a temporary (reset) or permanent (clear) connection failure has occurred. It is known as the connection failure indication operation. Finally, the X'0000' operation indicates incoming data. It is known as the receive data operation.

## Connections

User-defined communications support allows X.25 connections over both switched and permanent virtual circuits. Your application program can have one or many connections active at once. They can be either SVC, PVC, or both. The Display Connection Status (DSPCANNSTS) command shows the state of the connection, logical channel identifier, virtual circuit type, and other information about the call. The states of the connection include activate pending, active, deactivate pending.



When the open connection request or call accept operations are not yet complete, the connection state is activate pending. Once the open connection request or call accept operations are complete with return and reason codes of zero, the connection state is active. When the close connection request is not yet complete, or if the connection is cleared by the network, but a close connection request has not been issued by the application program, the connection state is deactivate pending.

**Notes:**

1. The connection enters the active state when the call accept packet is sent on the network, which is independent of the application program receiving the results of the open connection request. Likewise, a connection can become completely closed (deactivated, and no longer appears on the DSPC>NNSTS screen) independent of the application program receiving the results of the close connection request. This closing occurs when the application confirms the connection failure.
2. A correctly encoded close connection request will always be successful. The only time a close connection request is not successful is when the application program has coded the close connection request incorrectly. See Using Connection Identifiers for more information.

## Connection Identifiers

To differentiate between connections, user-defined communications support and an application program both use connection identifiers from the time the connection is started to the time the connection has successfully ended.

User-defined communications support assigns an identifier for each connection. This identifier is reported back to your application program as the provider connection end point (PCEP). In the same manner, your application program assigns an identifier for each connection and reports it to the communications support as the user connection end point (UCEP). This exchange of identifiers allows both the communications support and the application program to refer to a connection in a consistent manner. The UCEP and PCEP are exchanged during the open connection request during the following operations:

- A PVC is opened.
- An outgoing call is requested.
- The call indication is received and the call accept is accepted.

User-defined communications support identifies a connection only in terms of PCEP and UCEP. For example, the user-defined communications support passes information to an application program and reports the UCEP to which the information pertains. In the same manner the application program initiates requests for a connection identified by the PCEP.

User-defined communications support uses PCEPs over again as they become free. PCEPs become free when the application program receives notification that the open connection request never completed successfully, or the close connection request completed successfully. This means that PCEPs are not used over again until the application calls the QOLRECV API, which returns either the open connection request or the close connection request. Until the PCEP is freed, the connection cannot be reused.

User-defined communications support places no restrictions on the value of the UCEP, and does not verify its uniqueness. Because user-defined communications passes all incoming data and connection failure indications to the application program using the UCEP connection identifier, the application should ensure uniqueness of each UCEP. See “Using Connection Identifiers” on page 251 for information on how to reuse connection identifiers.

## Connection Information

In order to ensure reliable end-to-end connectivity, an application program must keep track of the control information for each connection it is responsible for. Some of this control information is shown in the following list.

- State of the connection (activating, active, deactivating, reset)

- PCEP for this connection
- SVC or PVC connection indicator
- Negotiated frame sizes; maximum data unit size
- Connection is no longer active indicator or state
- Other application specific information


The application program can use the UCEP as an index into the program's data structures, which keep track of this control information.

## Switched Virtual Circuit (SVC) Connectivity

### Configuration

All the users of an X.25 line description share the SVCs that are configured for that line description. These users are SNA, asynchronous X.25, OSI, TCP/IP, and user-defined communications. You should define the line description with enough SVCs to accommodate all of the users of the X.25 line.

Any SVCs defined in the X.25 line description that are not in use by any controllers (including the network controller) are available to an application program. The available SVCs are distributed as they are requested by the users of the X.25 line description.

See the X.25 Network Support  book for more information on configuring X.25 line descriptions.

For user-defined communications, the application uses an SVC when it either initiates a call, or receives an incoming call. The SVC is no longer in use when the application successfully initiates a clear request to the SVC. Like PVCs, SVCs allow only one application program to have an active connection using the virtual circuit at a time.

## Inbound Routing Information

Before an application program can receive and accept an incoming call, it must first describe to the user-defined communications support the X.25 calls that should be routed to the application. The application does this by issuing a program call to the QOLSETF API, specifying the inbound routing information in the filter.

The inbound routing information that an application program specifies is the first byte of call user data called the protocol ID, or the protocol ID combined with the calling DTE address. In addition, the application specifies whether it will accept calls with fast select and reverse charging indicated. The application program can either accept or reject any calls of which it receives indications. The advantage of using filters to allow the system to reject some calls (based on protocol ID, calling DTE address, fast select, and reverse charging indicated in the incoming call) is that the application is relieved of some of the calls it would always reject.

Once the connection is active, data flows end-to-end between systems and does not need any other technique to route it to the appropriate application.

## End-to-End Connectivity

End-to-end connectivity is achieved when one system initiates a call and another accepts the call. When this happens, a connection is established, and the state of the connection is active. It remains active until either one of the application programs initiates a clear request, or the network (or system) clears the connection due to an error condition.

## Permanent Virtual Circuit (PVC) Connectivity

### Configuration

SNA and asynchronous X.25 controllers use PVCs on the X.25 line by configuring the controller description to logically attach to the PVC. This is not true for users of the network controller description. When a PVC is in use by an application program, the system will logically attach the network controller to the PVC. This means that any PVC defined in the X.25 line description and not attached to any controller (including the network controller) is available for use by any application that has a link enabled for the network to which the line is attached.

Because the attaching of PVCs to applications is programmable, one job can have an open connection over the PVC, end the connection, and then another job can open a different connection over the same PVC. Like SVCs, PVCs allow only one application program at a time to have an active connection using the virtual circuit.

## **Inbound Routing Information**

By definition, the PVC does not require a call to set up a path from one system to another system. As its name suggests, this path always exists (permanent). Because there is no incoming call to route, the application has no need to set a filter for the inbound routing information. Once the application has opened the PVC, there is no other information needed for the system to route packets on the PVC to the application.

## **End-to-End Connectivity**

The application is responsible for opening and closing PVC connections. To open a PVC, the application uses the open connection request operation, just as it does to initiate an X.25 SVC call. To close the PVC, the application uses the close connection request just as it does to clear the SVC call.

Both systems that want to communicate end-to-end must first open the virtual circuit on the local system. When the PVC is opened on the iSeries server it is considered active and in use by the application. This is true even if the corresponding remote system doesn't have the virtual circuit active. On the iSeries server, an open connection request always completes with return and reason codes of zero as long as the PVC is defined in the line description and is not in use by another application. There is no way to detect whether true end-to-end connectivity exists on the PVC.

If the virtual circuit is not active on both systems, and one system attempts to communicate with the other, the virtual circuit on the system with the open PVC connection is reset. An application that supports X.25 resets, sees the reset arrive as a result of the attempt to send data. In order to continue, the application responds to the reset. An application that does not support X.25 resets sees a connection failure. The application closes the PVC and opens the PVC again in order to continue to use the PVC.

Similarly, when a PVC connection is closed from one system, the other system sees a reset (if reset is supported by that application) or a connection failure if reset is not supported. If the application sees a reset, it must respond to the reset before communications can continue on that connection.

## **Sending and Receiving Data Packets**

### **Data Sizes**

Data units larger and smaller than the negotiated transmit packet size can be sent by an application program. Each data unit will be segmented into the appropriate packet sizes by the iSeries server. Contiguous data larger than the negotiated packet size can also be sent. The data is divided into individual packets and sent out with the more-data indicator on. The application program should request that the data unit size be a multiple of the transmit and receive packet sizes configured in the line description. The application program sets other important values that pertain to each connection. See "X.25 SVC and PVC Output Operations" on page 43 for information about these values.

The values your application supplies should be carefully determined and tailored to the needs of the application. Similarly, your application uses the values returned from the system to ensure that the application does not exceed negotiated limitations.

The application uses three values to determine how to fill the user-space output buffer. These values are:

- Data unit size
- Maximum data unit assembly size
- Negotiated transmit packet size

The data unit size is the value that an application specifies when the link is enabled. The maximum data unit assembly size is the total length of contiguous input data that is assembled by the iSeries server before passing it to the application. Contiguous data units have the more-data indicator set on in each descriptor for all the data units in the sequence except the last data unit, which has the more-data indicator set off. The application specifies the maximum data unit assembly size on the open connection request. The maximum data unit assembly size should always be greater than the data unit size to make full use of the user spaces. The negotiated transmit packet size is returned when the open connection request completes. The application uses these values together to determine how to fill in the user space output buffer.

**Note:** If the maximum data unit assembly size is exceeded, the data is passed up to the application with the more-data indicator on. If the connection is abnormally reset or cleared, the application may not receive the rest of the contiguous data, which was in progress during the connection failure.

If the two applications remain without exceeding the maximum data unit assembly size supported on the remote system, the system guarantees that the application receives the complete, contiguous data packet sequence.

See Maximum Amount of Outstanding Data (page 277) for related information on incoming data limitations.

## Interrupts

The interrupt is a special data packet. The X.25 network imposes the restriction that a DTE cannot have more than one outstanding interrupt on any virtual call in each direction. An application program issues an interrupt by calling the `qolsend` API. The `qolsend` API does not return to the application program until the interrupt confirmation has been received. It is important to understand the interrupt confirmation procedures of the remote DTE and its implications to the local system and application.

## Flow Control

The iSeries server sends the Receive Ready (RR) and Receive Not Ready (RNR) packets on behalf of the application program. The distribution of these packets is based on the automatic flow control field in the open connection request operation. The automatic flow control (RR/RNR) is sent to prevent one system from overrunning another system with data.

When the automatic flow control value is exceeded for a connection because a remote system is sending data at a rate too fast for the local system, an RNR packet is sent on behalf of the application on that local system. Once the application on the local system receives the data, an RR is sent to allow more data to be received by the local system's communications support.

The automatic flow control value should be set high enough so that RR/RNR processing does not affect performance on the virtual circuit, and low enough that the application can process the data fast enough. If an application is coded properly, the RR and RNR processing is not noticed by the application, just as for other system users of X.25.

To avoid situations where the virtual circuit is not operational because an RNR was sent, or to avoid excessive amounts of RR and RNR processing, the application program should always attempt to receive all the data from the communications support. An application that is not coded correctly can cause another application to wait indefinitely for an RR to open the virtual circuit for communications. When the applications are coded correctly, the RR and RNR packet sequences are not noticed by the applications.


### Maximum Amount of Outstanding Data

The communications support sets aside a limited amount of data for the applications it is servicing. For X.25, it is 128K for each connection. If the 128K limitation is met, an error log entry is created and the connection is cleared (SVCs) or reset (PVCs) by the system. Before this limit is reached, the iSeries server attempts to slow the incoming data traffic by issuing an RNR on behalf of the application. An RR is sent after the application has received one-third of the amount of outstanding data.

### Reset Support

When an application program initiates a reset, it is also responsible for discarding any data that the user-defined communications support has received. The user-defined communications support only discards data if the connection is closed.

## X.25 Call Control

The X.25 support routes X.25 calls arriving to the iSeries server primarily based on the protocol ID field. This field is the first byte of call-user data in the X.25 call packet. For more information on the X.25 support, see the X.25 Network Support  book.

## Performance Considerations

The X'0000' operation is completely synchronous. This means that control is not returned to the application until all the data passed in the data units are sent and confirmed by the DCE. Some implications of this are:


- If the application sends data on a connection that has data flow turned off (the remote system sent an RNR to the local iSeries server), a subsequent call to the qolsend API with operation X'0000' will not return until the remote system sends the RR to turn flow back on for the connection.
- When transmitting Interrupt packets, control is not returned to the application until the interrupt is confirmed by the remote DTE. If the remote DTE is an iSeries server, the interrupt is confirmed by the iSeries server X.25 packet layer support. If the network is congested, the use of Interrupt packets may cause a decrease in performance for the application.

In these situations, it may be appropriate to have one job for each connection (each virtual circuit).

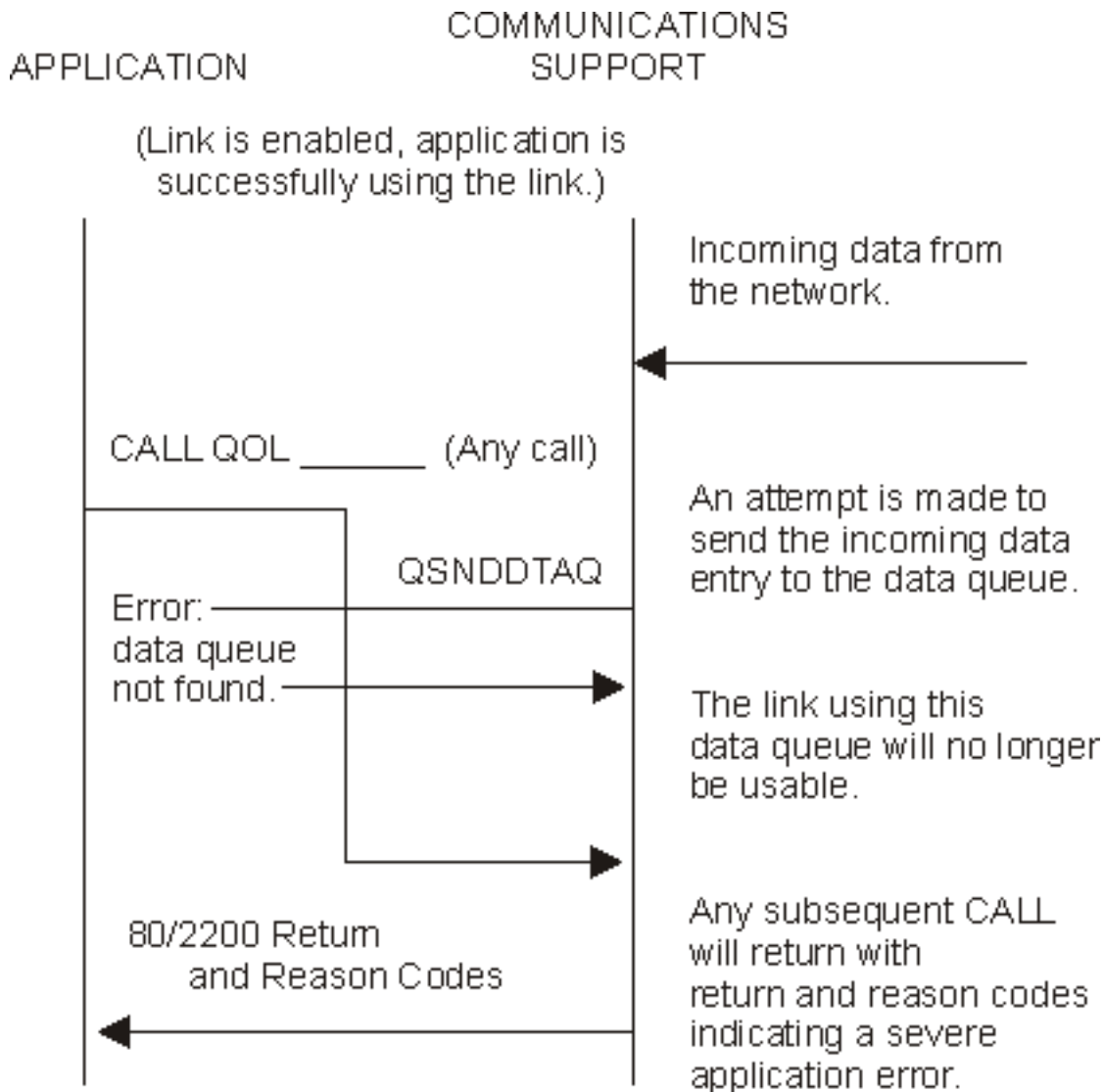
---

## Queue Considerations

An application program uses a data queue or user queue for communications between the application and the communications support. The application should create the queue prior to the call to the

QOLELINK API. For more information on creating and using a queue, see the CL Programming  book. The link will never be fully enabled if the queue does not exist. For example, in Figure 1-16 (page 277), communications is no longer available when the user-defined communications support detects that the data queue has been deleted. The same is true for user queues.

**Figure 1-16. Using the Data Queue**



In addition to using a queue for communications between the application and the user-defined communications support, the application can use the queue to provide communications with other applications.

If multiple processes are using the same queue, the queue can be manipulated so that each process receives queue entries based on the unique key for each application. This allows the jobs to put many kinds of entries on the queue. For example, one key value is used for communications between the application and the system, and another key value is used for communications between the user-defined communications applications and other applications. Key values can also serve as a way to prioritize entries on the queue.

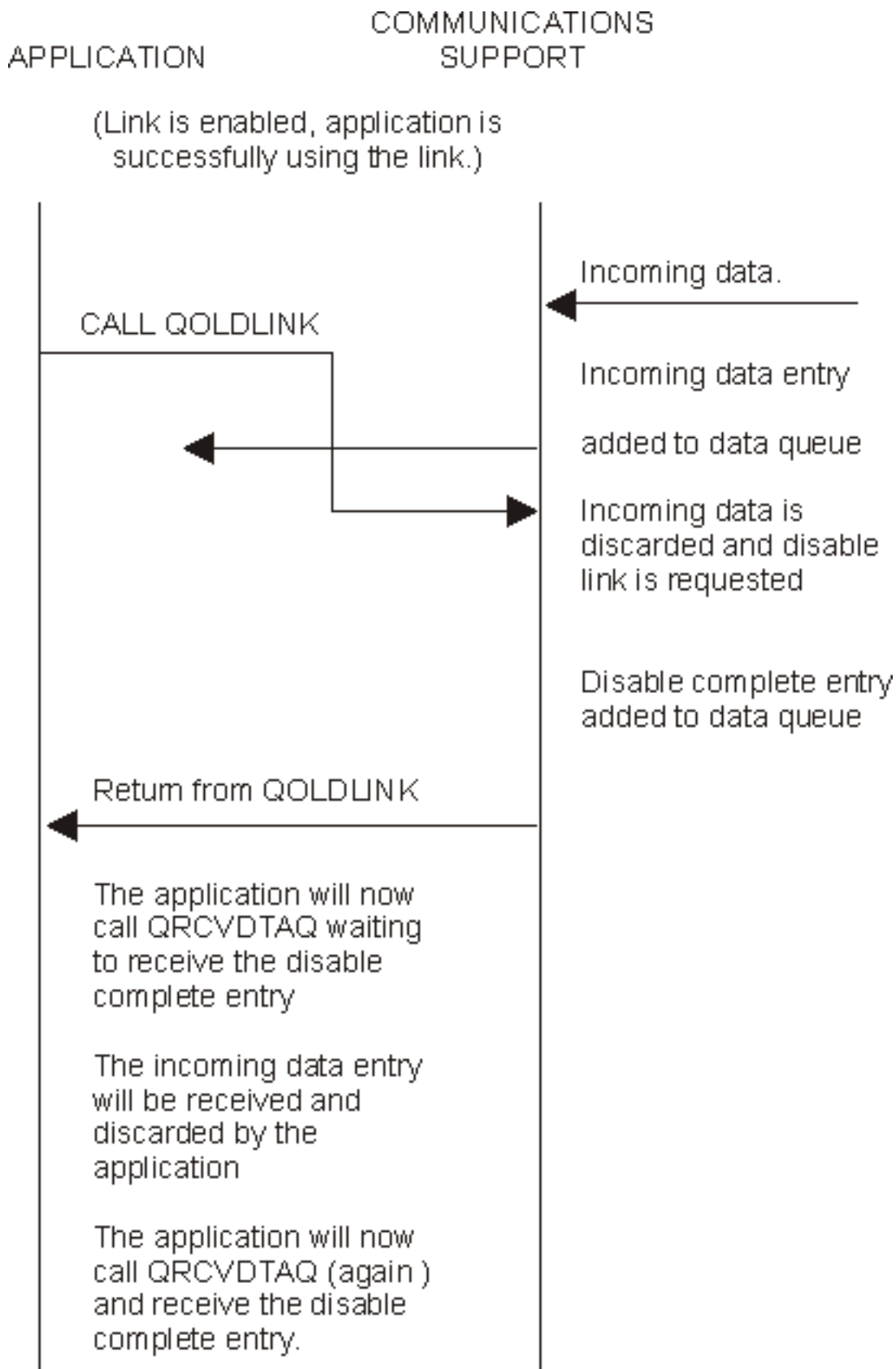
The content of the queue entries that the application defines and uses is not restricted by the user-defined communications support. User-defined communications support never attempts to receive any entries from the queue. It is the responsibility of the application to receive the entries from the queue and perform the appropriate actions for an entry based on its handle (or timer handle).

This means that it might be necessary for the application to clear the old messages from the queue if it has been used. For example, if a link is disabled, all communications entries for that link (denoted by the communications handle) prior to the disable complete entry are no longer valid.

**Note:** Timer support does not depend on the user-defined communications support; therefore, timer entries are still valid.

The following example shows an incoming data entry that the application receives is no longer valid because the application made a request to disable the link.

**Figure 1-17. Application Disables the Link**





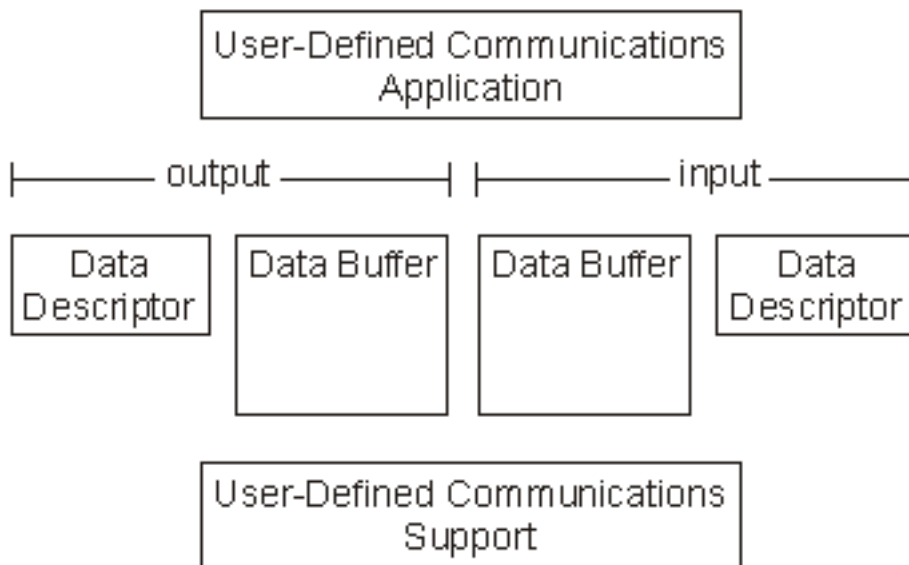
---

## User Space Considerations

Your application uses user space objects (\*USRSPC) to hold the input and output buffers and descriptors. The iSeries server provides APIs you can use to manipulate the user spaces.

When you use the user-defined communications support, you create the user spaces, a total of four, as part of an enable link request (the QOLELINK API). For each link, there is an input buffer, input buffer descriptor, output buffer, and output buffer descriptor. The buffers and descriptors are used to pass information to and from your application program. The buffers are used to contain user data. The descriptors are used to describe the data (length and other qualifiers). If the enable link request is not successful (return and reason codes are nonzero), the user spaces are not created.

Figure 1-18. User Spaces



The buffers are divided into equally sized, contiguous sections called data units. The output buffer contains data to be sent on the network. The input buffer contains data received from the network. The size of each data unit, as well as the number of data units created, is returned from the QOLELINK API when the link is enabled.

The buffer descriptors are divided into equally sized, contiguous sections called descriptor elements. Each descriptor element describes the data in the corresponding data unit of the buffer. For example, descriptor element 1 describes the data in data unit 1 of the buffer. The size of each descriptor element is 32 bytes.

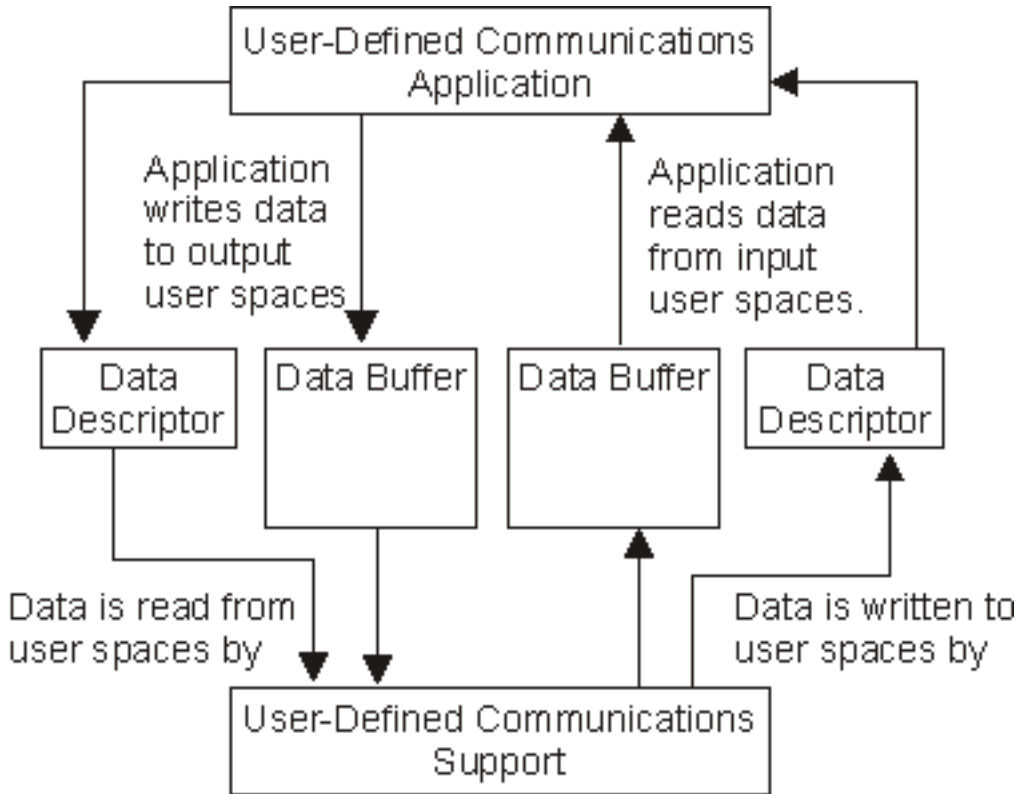
For complete and specific information about the input/output buffers, descriptors, data units, and data elements, see the sections in "User-Defined Communications Support APIs" on page 1 describing the individual APIs.

Your application provides the library and name of the user space object that is created. The descriptive text for the object always contains the name of the job that is using these spaces. Finally, when the link is disabled (either explicitly or implicitly), these user spaces are deleted by the user-defined communications support. See "Disable Link (QOLDLINK) API" on page 2 for more information on disabling the link.

The application reads from the input buffer and descriptor, and writes to the output buffer and descriptor. Similarly, the user-defined communications support reads from the output buffer and descriptor and writes to the input buffer and descriptor. As soon as the call to the qolsend API or the QOLRECV API is complete, the application can access these user spaces.

If changes or deletions to the user spaces occur while they are in use by the user-defined communications support, a severe application error is reported to the application, and communications over the link associated with the user spaces is no longer possible.

**Figure 1-19. Input/Output Operations**



The user-defined communications support defines logical views for the user spaces. These views are sometimes called formats. There is a format for filters, sending and receiving LAN frames, and sending and receiving X.25 packets. See “Send Data (QOLSEND) API” on page 38 and “Receive Data (QOLRECV) API” on page 21 for details on these formats.

Your application must set all the data fields required for the format. There are two types of byte fields in the buffer and descriptors, character (CHAR) and binary (BIN). Binary implies that the value is used as a numeric value. Sometimes this might be a 1-byte numeric value; for example, 12 = X'0C'. If you write the application in a language that is not capable of setting this type of binary field, the field should be declared as character and set to X'0C'. The character type contains an EBCDIC value, printable or not printable. In contrast, all parameter values are either character or 4-byte binary. See “Programming Languages” on page 250 for help in writing your application so that it can provide the expected input for the user-defined communications support.

The communications support never changes the output buffer; therefore, your application is responsible for initializing the buffer and descriptor for the next operation, if necessary. The data in the output buffer can also be used to help determine why a particular operation is not successful.

For performance reasons, your application should attempt to fill the output buffer as completely as possible.

Finally, for security reasons, your application chooses the library the user space object will reside in. The library can be any system library, including QTEMP. The advantage (or disadvantage) of using QTEMP for user space objects is that only the job which has enabled the links has access to the user spaces. This is

because a QTEMP library exists for each job on the system. If the user space objects are in any other library, any job having authority to the library that the user spaces are in can access them.

---

## Return Codes and Reason Codes

When control returns from a user-defined communications API to your application program, the status of the operation is located in the reason code and return code output parameters for each API.

Return codes are 4-byte numbers that determine the recovery action to take. They are grouped into the following categories:

- 00 — Operation successful, no recovery needed
- 80 — Irrecoverable error, need to disable link
- 81 — Irrecoverable error, do not need to disable link
- 82 — Recoverable error, enable link failed
- 83 — Recoverable error, see recovery actions

Reason codes are 4-byte numbers that determine what error occurred. They are grouped into the following categories:

- 0000 — No error
- 1xxx — Parameter validation or format error
- 20xx — Line, controller, or device description error
- 22xx — Data queue error
- 24xx — Buffer or buffer descriptor error
- 30xx — Link state error
- 32xx — Connection state error
- 34xx — Timer state error
- 4xxx — Communication error
- 8xxx — Application error
- 9999 — A condition in which an Authorized Program Analysis Report (APAR) may be submitted

**Note:** 'x' represents any decimal number. For example, 1xxx represents the range 1000 - 1999.

For complete and specific information about the reason codes and return codes, see the sections in "User-Defined Communications Support APIs" on page 1 describing the individual APIs.

---

## Messages

The following messages are used to signal the success or failure of operations performed by the user-defined communications APIs:

- **Information**

Message ID	Error Message Text
CPI91F0 I	X.25 network error occurred.
CPI91F1 I	ISDN network error occurred.

- **Escape**

Message ID	Error Message Text
CPF91F0 E	Internal system error.
CPF91F1 E	User-defined communications application error.

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

- **Diagnostic**

Message ID	Error Message Text
CPD91F0 D	Error detected in program &1. Condition code is &2.
CPD91F1 D	Unexpected error detected in program &1. Condition code is &2.
CPD91F2 D	User space &1 or &3 not accessible.
CPD91F3 D	Data limit exceeded. Some data not sent.
CPD91F4 D	Error while accessing queue &1 in library &2.
CPD91F5 D	Error while accessing queue. Time &1 canceled.
CPD91F6 D	Error occurred on line &1 while in use.
CPD91F7 D	Recovery canceled for network interface &3 or line &1.
CPD91F8 D	Error while accessing queue &1 in library &2.
CPD91F9 D	Error while enabling line &1.

Top | “Communications APIs,” on page 1 | APIs by category

---

## Configuration and Queue Entries

- Configure user-defined communications support
- Set up the entries that user-defined communications support can send to the queue

## Configuring User-Defined Communications Support

This section describes what needs to be configured before your application program can use the user-defined communications APIs. You can either use the system-supplied menus or the Control Language (CL) commands to do this configuration. For more information on using queue APIs, see the Object APIs in the Information Center.

## Links

Links allow your application program to use a token-ring, Ethernet, FDDI, wireless, or X.25 communications line. A link is made up of the following communications objects:

- Token-ring, Ethernet, FDDI, wireless, or X.25 line description
- Network controller description
- Network device description of type \*USRDFN
- Network interface description for ISDN (X.25 only)

You need to configure the line description; user-defined communications support automatically configures a network controller and a network device description of type \*USRDFN when the link is enabled. If you are using X.25 over ISDN, the network interface description must also be configured. The network interface, line, controller, and device descriptions are automatically varied on, if necessary.

Use the following commands to create or change line descriptions:

- CRTLINDDI — Create Line Description (DDI)
- CHGLINDDI — Change Line Description (DDI)
- CRTLINETH — Create Line Description (Ethernet)
- CHGLINETH — Change Line Description (Ethernet)
- CRTLINTRN — Create Line Description (Token-Ring)
- CHGLINTRN — Change Line Description (Token-Ring)

- CRTLINWLS — Create Line Description (Wireless)
- CHGLINWLS — Change Line Description (Wireless)
- CRTLINX25 — Create Line Description (X.25)
- CHGLINX25 — Change Line Description (X.25)

Use the following commands to create or change controller descriptions:


- CRTCTLNET — Create Controller Description (Network)
- CHGCTLNET — Change Controller Description (Network)

Use the following commands to create or change device descriptions:

- CRTDEVNET — Create Device Description (Network)
- CHGDEVNET — Change Device Description (Network)

Use the following commands to create or change network interface descriptions:

- CRTNWIISDN — Create Network Interface Description (ISDN)
- CHGNWIISDN — Change Network Interface Description (ISDN)

See the Communications Configuration  book on the V5R1 Supplemental Manuals Web site for more information on configuring communications.

## Queue

User-defined communications support uses a queue to inform your application program of some action to take or of an activity that is complete. You must create the queue before the link is enabled.

The size of each queue entry must be large enough to accommodate the user-defined communications support entries. See the following “Queue Entries” for more information on the entries that user-defined communications support can send to the queue.

Use the Create Data Queue (CRTDTAQ) command to create your data queues. Use the QUSCRTUQ and QUSDLTUQ APIs to create and delete your user queues.

---

## Queue Entries

This section describes the entries user-defined communications support can send to the queue.

### General Format

The length of each entry is always at least 80 bytes. When using a keyed queue, however, each entry can be as large as 336 bytes, depending on the size of the key value supplied to the user-defined communications support.

Table 1 (page 285) shows the general format of each user-defined communications support entry.

**Table 1. Queue Entry General Format**

Entry type Char(10)	Entry ID Char(2)	Entry data Char(68)	Key CHAR(256)
Bytes 1-10	11-12	13-80	81-336

### Entry type

This indicates the type of entry on the queue and will be \*USRDFN for all user-defined communications support entries.

## Entry ID

This uniquely identifies each entry within an entry type. User-defined communications support has five entries defined:

- Enable-complete entry (entry ID = '00')
- Disable-complete entry (entry ID = '01')
- Permanent-link-failure entry (entry ID = '02')
- Incoming-data entry (entry ID = '03')
- Timer-expired entry (entry ID = '04')

**Note:** The entry type of \*USRDFN and all associated entry IDs, either defined or undefined, are reserved for user-defined communications support. Therefore, your application program should not define entries using this entry type.

## Entry data

This data is useful to your application program and varies according to the entry ID.

## Key

When using a keyed queue, this is the key value supplied to the user-defined communications support.

## Enable-Complete Entry

The enable-complete entry is sent to the queue when the enable link operation is complete. This entry is only sent after the Enable Link (QOLELINK) API returns to your application program with a successful return and reason code.

**Note:** The QOLELINK API only initiates the enabling of the link. Your application program must wait for the enable-complete entry before attempting to perform input or output on the link.

Table 2 (page 286) shows the format of the enable-complete entry.

**Table 2. Enable-Complete Entry**

*USRDFN	'00'	Communications handle	Status	Reserved	Key
Bytes 1-10	11-12	13-22	23	24-80	81-336

## Communications handle

The name of the link that is being enabled. Your application program supplies this name when the QOLELINK API is called.

## Status

This indicates the outcome of the enable link operation. A character value of zero indicates the enable link operation was successful and I/O is now possible on this link. A character value of one indicates the enable link operation was not successful (the job log contains messages indicating the reason). The user-defined communications support disables the link when the enable link operation does not complete successfully and the disable-complete entry is not sent to the queue.

## Key

The key value associated with the enable-complete entry when using a keyed queue. Your application program supplies this key value when the QOLELINK API is called. When using a non-keyed queue (indicated by supplying a key length of zero to the QOLELINK API) this field is not present.

## Disable-Complete Entry

The disable-complete entry is sent to the queue when a link is successfully disabled. This entry is always the last entry sent by the user-defined communications support on this link and, therefore, provides a way for your application program to remove any enable-complete, incoming-data, or permanent-link-failure entries previously sent to the queue.

**Note:** User-defined communications support does not associate timers with links. Therefore, it is possible for a timer-expired entry to be sent to the queue after the link is disabled. Your user-defined communications application program is responsible for handling this.

Table 3 (page 287) shows the format of the disable-complete entry.

**Table 3. Disable-Complete Entry**

*USRDFN	'01'	Communications handle	Reserved	Key
Bytes 1-10	11-12	13-22	23-80	81-336

### Communications handle

The name of the link that has been disabled. Your application program supplies this name when the QOLELINK API is called to enable the link.

### Key

The key value associated with the disable-complete entry, when using a keyed queue. Your application program supplies this key value when the QOLELINK API is called to enable the link. When using a non-keyed queue (indicated by supplying a key length of zero to the QOLELINK API) this field is not present.

## Permanent-Link-Failure Entry

The permanent-link-failure entry is sent to the queue when error recovery is canceled on a link. You must disable and then enable the link to recover.

Table 4 (page 287) shows the format of the permanent-link-failure entry.

**Table 4. Permanent-Link-Failure Entry**

*USRDFN	'02'	Communications handle	Reserved	Key
Bytes 1-10	11-12	13-22	23-80	81-336

### Communications handle

The name of the link on which the failure has occurred. Your application program supplies this name when the QOLELINK API is called to enable the link.

### Key

The key value associated with the permanent-link-failure entry, when using a keyed queue. Your application program supplies this key value when the QOLELINK API is called to enable the link. When using a non-keyed queue (indicated by supplying a key length of zero to the QOLELINK API) this field is not present.

## Incoming-Data Entry

The incoming-data entry is sent to the queue when the user-defined communications support has data for your application program to receive. Your application program should call the Receive Data (QOLRECV) API to pick up the data when this entry is received.

**Note:** Another incoming-data entry is not sent to the queue until your application program picks up all the data from the user-defined communications support. The data available parameter on the call to the QOLRECV API indicates that the receipt of data is not complete.

Table 5 (page 288) shows the format of the incoming-data entry.

**Table 5. Incoming-Data Entry**

*USRDFN	'03'	Communications handle	Reserved	Key
Bytes 1-10	11-12	13-22	23-80	81-336

### Communications handle

The name of the link on which the data has come in. Your application program supplies this name when the QOLELINK API is called to enable the link.

### Key

The key value associated with the incoming-data entry, when using a keyed queue. Your application program supplies this key value when the QOLELINK API is called to enable the link. When using a non-keyed queue (indicated by supplying a key length of zero to the QOLELINK API) this field is not present.

## Timer-Expired Entry

The timer-expired entry is sent to the queue when a timer, previously set by your application program, ends.

Table 6 (page 288) shows the format of the timer-expired entry.

**Table 6. Timer-Expired Entry**

*USRDFN	'04'	Timer handle	User data	Key
Bytes 1-10	11-12	13-20	21-80	81-336

### Timer handle

The name of the expired (ended) timer. Your application program returns this name when the Set or Cancel Timer (QOLTIMER) API is called to set the timer.

### User data

The data associated with the expired timer. Your application program supplies this data when the QOLTIMER API is called to set the timer.



## Key

The key value associated with the timer-expired entry, when using a keyed queue. Your application program supplies this key value when the QOLTIMER API is called to set the timer. When using a non-keyed queue (indicated by supplying a key length of zero to the QOLTIMER API) this field is not present.

Top | “Communications APIs,” on page 1 | APIs by category

---

## Debugging of User-Defined Communications Applications

This section is intended to help you debug your user-defined communications applications. It contains information about:

- System services and tools
- Error codes reported to the application program and QSYSOPR operation
- Common error list

### System Services and Tools

There are several tools on the iSeries<sup>(TM)</sup> server you can use to debug your user-defined communications application. Some of the system provided tools that are useful for developing user-defined communications applications include the following CL commands:

- Program Debug (STRDBG)
- Work with Job, Work with Communications Status (WRKJOB OPTION(\*CMNSTS))
- Work with Job, Display Job Log (WRKJOB OPTION(\*JOBLOG))
- Display Connection Status (DSPCNNSTS)
- Display Inbound Routing Data (press F6 (Display inbound routing information) following the DSPCNNSTS command)
- Check Communications Trace (CHKCMNTRC)
- Delete Communications Trace (DLTCMNTRC)
- End Communications Trace (ENDCMNTRC)
- Print Communications Trace (PRTC MNTRC)
- Start Communications Trace (STRCMNTRC)
- Start System Service Tools (STRSST)
  - Work with communications trace
  - Work with error log
- Dump System Object (DMPSYSOBJ)

### Program Debug

Program debug (STRDBG) allows you to trace the program and variables, set stops, change variables, and display variables. You can use this function to verify that the parameters are passed correctly.

### Work with Communications Status

The Work with Job command, Work with Communications Status option, (WRKJOB OPTION(\*CMNSTS)) shows the enabled links and operation counts for each link. It also reports information such as the communications handle the last operation requested, and the total input, output, and other operations requested. This information is shown for every link enabled by the job.

## Display Job Log

The Work with Job command, selecting the Display job log option (WRKJOB OPTION(\*JOBLOG)) allows you to view the messages in the job log that help determine the exact cause of the problem.

## Display Connection Status

The Display Connection Status (DSPCNNSTS) command shows information about the switched virtual circuits (SVCs) and permanent virtual circuits (PVCs) that are in use by the application using the device description.

**Note:** The Display Line Description (DSPLIND) command also shows for each line, the SVCs that are in use, available, or attached to a controller description. This is not true for PVCs.


## Display Inbound Routing Information

Pressing F6 (Display inbound routing information) when the Display Connection Status display is shown (DSPCNNSTS command) shows the results of the calls to the Set Filter (QOLSETF) API. It also helps to determine which device description has set a filter with duplicate inbound routing information.

## Work with Communications Trace

Using the communications trace function you can obtain information about a communications line. You can access the communications trace function through the following CL commands:

- Check Communications Trace (CHKCMNTRC)
- Delete Communications Trace (DLTCMNTRC)
- End Communications Trace (ENDCMNTRC)
- Print Communications Trace (PRTC MNTRC)
- Start Communications Trace (STRCMNTRC)

For more information on using the communications trace CL commands, see the Communications Management  book.

You can also access the communications trace function through the system service tools. You can use this function by entering the Start System Service Tools (STRSST) command and selecting the option to start a service tool.

Using the option to Work with communications trace shows data just as it appears to the network. If the application requests that data be sent and the request does not appear in the communications trace, the request is rejected. The return and reason codes, and the error code reported in the parameter list for the Send Data (QOLSEND) API indicates the reason the request was rejected.

## Work with Error Log

The error log utility is part of the system service tools. You can use it by entering the Start System Service Tools (STRSST) command and selecting the option to start a service tool.

Some communications errors are reported by the system to the error log. A remote application that is communicating with a user-defined communications application on the local system, could cause an entry to be generated in the error log if one of the following conditions are met:

- When using a LAN, data is not received by the application and exceeds internal threshold values (3 MB).
- When using an X.25 network, data is not received by the application and exceeds internal threshold values (128KB).

For both cases, the associated message in QSYSOPR identifies the error log that contains the error log entry. The error log entry contains information only.

## Dump System Object to View User Spaces

The Dump System Object (DMPSYSOBJ) command is used to inspect the user spaces after they are filled in by your application. The following examples indicate what the user spaces look like for some of the operations.

### User Space to Set a Filter to Route Inbound Data

This user space is filled in to activate two X.25 filters which will route any X.25 call containing X'BB', or X'DD' in the first byte of call user data (protocol ID byte).

Figure 1-1. User Space to Set a Filter to Route Incoming X.25 Calls

```

5738SS1 V2R1M0 910524          OS/400 DUMP          006625/QSECOFR/QPADEV0001    12/21/90 12:42:07    PAGE 1
DMPSYSOBJ PARAMETERS
OBJ- OUTBUFFER                  CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-          SPACE          *USRSPC
NAME-      OUTBUFFER          TYPE-      19  SUBTYPE-      34
LIBRARY-   USRDFNCMN         TYPE-      04  SUBTYPE-      01
CREATION-  12/21/90 12:40:03    SIZE-     00002200
OWNER-     QSECOFR           TYPE-      08  SUBTYPE-      01
ATTRIBUTES-      0800          ADDRESS-   00A00A00  0000
SPACE ATTRIBUTES-
000000  00000080 00000060 1934D6E4 E3C2E4C6 C6C5D940 40404040 40404040 40404040 * - OUTBUFFER *
000020  40404040 40404040 E0000000 00000000 00002000 00800000 00000000 00000000 * \ *
000040  00000000 00000000 0005004D 42000400 00000000 00000000 00000000 00000000 * (a *
SPACE-
000000  01000002 001001BB 00000000 00000000 00000000 000001DD 00000000 00000000 * Y t *
000020  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
LINES 000040 TO 001FFF SAME AS ABOVE
POINTERS-
NONE
OIR DATA-
TEXT-
000000 D8D7C1C4 C5E5F0F0 F0F1D8E2 C5C3D6C6 D9404040 F0F0F6F6 F2F5 *QPADEV0001QSECOFR 006625 *
SERVICE-
000000  40404040 40404040 40404040 40404040 40404040 40F14040 40404040 40404040 * 1 *
000020  40404040 40404040 404040E5 F2D9F1D4 F0F0F9F0 F1F2F2F1 F1F2F4F0 F0F44040 * V2R1M00901221124004 *
000040  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 * *
000060  40404040 40404040 40404040 40404040 40404040 40404040 00000000 00000000 * *
000080  00000000 00000000 00000000 00000000 00000000 00000000 40400000 00000000 * *
0000A0  00000000 00000000 * *
END OF DUMP
***** END OF LISTING *****

```

### User Space for X'B000' Operation, Initiating an SVC Call

The user space below has been filled in to initiate an SVC call specifying the following:

- Default packet and window sizes
- D-bit (not selected)
- Reverse charging (not selected)
- Fast select (not selected)
- Closed user group (not selected)
- Other facilities (not selected)
- One byte of call user data, X'BB', which is the protocol identifier
- X.25 reset not supported by the user-defined communications application program
- 16KB is the maximum amount of contiguous data to be received
- Automatic flow control value of 32

Figure 1-2. User Space to Send an SVC Call

```

5738SS1 V2R1M0 910524          OS/400 DUMP          006625/QSECOFR/QPADEV0001  12/21/90 12:47:42          PAGE 1
DMPYSOBY PARAMETERS
OBJ- OUTPUTBUF                  CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-                     SPACE                      *USRSPC
NAME-      OUTPUTBUF            TYPE-          19  SUBTYPE-      34
LIBRARY-   USRDFNCMN           TYPE-          04  SUBTYPE-      01
CREATION-  12/21/90 12:36:28   SIZE-          00001200
OWNER-     QSECOFR             TYPE-          08  SUBTYPE-      01
ATTRIBUTES- 0800              ADDRESS-       00A00100 0000
SPACE ATTRIBUTES-
000000 00000080 00000060 1934D6E4 E3D7E4E3 C2E4C640 40404040 40404040 40404040 * - OUTPUTBUF *
000020 40404040 40404040 E0000000 00000000 00001000 00800000 00000000 00000000 * \ *
000040 00000000 00000000 0005004D 42000400 00000000 00000000 00000000 00000000 * (a *
SPACE-
000000 02000000 FFFFFFFF FFFFFFFF 00000000 00000008 40100001 00000000 00000000 * *
000020 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
      LINES 000040 TO 0000BF SAME AS ABOVE
0000C0 00000000 00000000 00000000 00000000 00000000 00000001 BB000000 00000000 * Y *
0000E0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
      LINES 000100 TO 0001BF SAME AS ABOVE
0001C0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00004000 * *
0001E0 00200000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
000200 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
      LINES 000220 TO 000FFF SAME AS ABOVE
POINTERS-
NONE
OIR DATA-
TEXT-
000000 D8D7C1C4 C5E5F0F0 F0F2D8E2 C5C3D6C6 D9404040 F0F0F6F6 F2F7 *QPADEV0002QSECOFR 006627 *
SERVICE-
000000 40404040 40404040 40404040 40404040 40404040 40F14040 40404040 40404040 * 1 *
000020 40404040 40404040 404040E5 F2D9F1D4 F0F0F9F0 F1F2F2F1 F1F2F3F6 F2F84040 * V2R1M00901221123628 *
000040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 * *
000060 40404040 40404040 40404040 40404040 40404040 40404040 00000000 00000000 * *
000080 00000000 00000000 00000000 00000000 00000000 00000000 40400000 00000000 * *
0000A0 00000000 00000000 * *
END OF DUMP
* * * * * E N D O F L I S T I N G * * * * *

```

User Space Containing an Incoming X.25 Call, Operation X'BB'

This user space shows the following:

- The call is using SVC 005
- Both transmit and receive packet sizes are 128
- Both transmit and receive window sizes are 7
- The calling DTE address is 40100000
- The called DTE address is 40200000
- No other facilities are requested
- One byte of call user data, X'BB', which is the protocol identifier

The application received this call because it had set a filter to indicate to the system that it should route incoming X.25 calls that have the first byte of call user data (the protocol identifier) equal to X'BB' to the application.

Figure 1-3. User Space Containing an Incoming X.25 Call

```

5763SS1 V3R1M0 940909      OS/400 DUMP      023099/QSYSOPR/QPADEV0014      03/07/94 11:57:24      PAGE 1
DMPYSOBY PARAMETERS
OBJ- INBUFFER              CONTEXT- USRDFNCMN
TYPE- *ALL SUBTYPE-*ALL
OBJECT TYPE-              SPACE
NAME-                      INBUFFER              TYPE-          19  SUBTYPE-        34
LIBRARY-                   USRDFNCMN              TYPE-          04  SUBTYPE-        01
CREATION-                  03/07/94 11:53:15    SIZE-         0000002200
OWNER-                     QSYSOPR              TYPE-          08  SUBTYPE-        01
ATTRIBUTES-                0800              ADDRESS-       000001DE7A00 0000
SPACE ATTRIBUTES-
000000 00FFFF00 00000060 1934C9D5 C2E4C6C6 C5D94040 40404040 40404040 40404040 * - INBUFFER *
000020 40404040 40404040 E0000000 00000000 00002000 00810000 00000000 00000000 * \ a *
000040 00000000 00000000 00D601DE 73000400 00000000 00000000 00000000 00000000 * 0 £ *
SPACE-
000000 00000005 00800007 00800007 00000000 00000008 40100000 00000000 00000000 * *
000020 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
LINES 000040 TO 0000BF SAME AS ABOVE
0000C0 00000000 00000000 00000000 00000000 00000000 00000001 BB000000 00000000 * *
0000E0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
LINES 000100 TO 00013F SAME AS ABOVE
000140 00000000 00000000 00000000 00000000 00000000 00000000 08402000 00000000 * *
000160 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
LINES 000180 TO 001FFF SAME AS ABOVE
POINTERS-
NONE
OIR DATA-
TEXT-
000000 D8D7C1C4 C5E5F0F0 F1F4D8E2 E8E2D6D7 D9404040 F0F2F3F0 F9F9 *QPADEV0014QSYSOPR 023099 *
SERVICE-
000000 40404040 40404040 40404040 40404040 40404040 40F14040 40404040 40404040 * 1 *
000020 40404040 40404040 40404040 404040E5 F3D9F1D4 F0F0F9F4 F0F3F0F7 F1F1F5F3 F1F54040 * V3R1M00940307115315 *
000040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 * *
000060 40404040 40404040 40404040 40404040 40404040 40404040 00000000 00000000 * *
000080 00000000 00000000 00000000 00000000 00000000 00000000 00000000 40400000 00000000 * *
0000A0 00000000 00000000 * *
USAGE-
000000 D8E2E8E2 D6D7D940 4040D9C3 C8C1E2F3 F2F0 *QSYSOPR RCHAS320 *
END OF DUMP
* * * * * E N D O F L I S T I N G * * * * *

```

### User Space to Accept an Incoming X.25 Call, Operation X'B400'

This user space was filled in to accept the incoming call, request default packet and window sizes, and no other additional facilities. The a maximum amount of contiguous data is set at 16KB and the automatic flow control is set at 32.

Figure 1-4. User Space to Accept an Incoming X.25 Call

```

5738SS1 V2R1M0 910524      OS/400 DUMP      006625/QSECOFR/QPADEV0001  12/21/90 12:48:06      PAGE  1
DMPYSOBY PARAMETERS
OBJ- OUTBUFFER              CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-                SPACE                *USRSPC
NAME-      OUTBUFFER        TYPE-          19  SUBTYPE-      34
LIBRARY-   USRDFNCMN        TYPE-          04  SUBTYPE-      01
CREATION-  12/21/90 12:40:03 SIZE-          00002200
OWNER-     QSECOFR          TYPE-          08  SUBTYPE-      01
ATTRIBUTES-      0800      ADDRESS-      00A00A00  0000
SPACE ATTRIBUTES-
000000  00000080 00000060 1934D6E4 E3C2E4C6 C6C5D940 40404040 40404040 40404040 * - OUTBUFFER *
000020  40404040 40404040 E0000000 00000000 00002000 00800000 00000000 00000000 * \ *
000040  00000000 00000000 0005004D 42000400 00000000 00000000 00000000 00000000 * (a *
SPACE-
000000  00000000 FFFFFFFF FFFFFFFF 00000000 00000000 00000000 00000000 00000000 * *
000020  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
      LINES 000040 TO 0001BF SAME AS ABOVE
0001C0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00004000 * *
0001E0  00200000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
000200  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
      LINES 000220 TO 001FFF SAME AS ABOVE
POINTERS-
NONE
OIR DATA-
TEXT-
000000  D8D7C1C4 C5E5F0F0 F0F1D8E2 C5C3D6C6 D9404040 F0F0F6F6 F2F5 *QPADEV0001QSECOFR 006625 *
SERVICE-
000000  40404040 40404040 40404040 40404040 40404040 40F14040 40404040 40404040 * 1 *
000020  40404040 40404040 404040E5 F2D9F1D4 F0F0F9F0 F1F2F2F1 F1F2F4F0 F0F44040 * V2R1M00901221124004 *
000040  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 * *
000060  40404040 40404040 40404040 40404040 40404040 40404040 00000000 00000000 * *
000080  00000000 00000000 00000000 00000000 00000000 00000000 40400000 00000000 * *
0000A0  00000000 00000000 * *
END OF DUMP
* * * * * E N D O F L I S T I N G * * * * *

```

## User Spaces for Sending Data, Operation X'0000'

Two user spaces are shown below. The first is the output buffer and the second is the output buffer descriptor.

The user spaces below are filled in to send three data units of 512 bytes each. The first two data units have the more data indicator turned on, indicating that all the data units are contiguous.

**Note:** This link was enabled, specifying a data unit size of 512 bytes.

**Figure 1-5. User Space (Buffer) to Send Three Data Units**

```

5738SS1 V2R1M0 910524      OS/400 DUMP      006625/QSECOFR/QPADEV0001  12/21/90 12:55:19  PAGE  1
DMPYSOBY PARAMETERS
OBJ- OUTPUTBUF              CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-                SPACE                                *USRSPC
NAME-      OUTPUTBUF        TYPE-      19  SUBTYPE-    34
LIBRARY-   USRDFNCMN        TYPE-      04  SUBTYPE-    01
CREATION-  12/21/90 12:36:28 SIZE-      00001200
OWNER-     QSECOFR          TYPE-      08  SUBTYPE-    01
ATTRIBUTES-      0800      ADDRESS-    00A00100  0000
SPACE ATTRIBUTES-
000000  00000080 00000060 1934D6E4 E3D7E4E3 C2E4C640 40404040 40404040 40404040 * - OUTPUTBUF *
000020  40404040 40404040 E0000000 00000000 00001000 00800000 00000000 00000000 * \ *
000040  00000000 00000000 0005004D 42000400 00000000 00000000 00000000 00000000 * (a *
SPACE-
000000  F0F10000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *01 *
000020  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
LINES 000040 TO 0001FF SAME AS ABOVE
000200  F0F20000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *02 *
000220  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
LINES 000240 TO 0003FF SAME AS ABOVE
000400  F0F30000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *03 *
000420  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
LINES 000440 TO 000FFF SAME AS ABOVE
POINTERS-
NONE
OIR DATA-
TEXT-
000000  D8D7C1C4 C5E5F0F0 F0F2D8E2 C5C3D6C6 D9404040 F0F0F6F6 F2F7 *QPADEV0002QSECOFR 006627 *
SERVICE-
000000  40404040 40404040 40404040 40404040 40404040 40F14040 40404040 40404040 * 1 *
000020  40404040 40404040 404040E5 F2D9F1D4 F0F0F9F0 F1F2F2F1 F1F2F3F6 F2F84040 * V2R1M00901221123628 *
000040  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 * *
000060  40404040 40404040 40404040 40404040 40404040 40404040 00000000 00000000 * *
000080  00000000 00000000 00000000 00000000 00000000 00000000 40400000 00000000 * *
0000A0  00000000 00000000 * *
END OF DUMP
* * * * * E N D O F L I S T I N G * * * * *

```

Figure 1-6. User Space (Descriptor Element) to Describe the Three Data Units

```

5738SS1 V2R1M0 910524          OS/400 DUMP          006625/QSECOFR/QPADEV0001    12/21/90 12:55:58    PAGE 1
DMPYSOBY PARAMETERS
OBJ- OUTPUTBUFDF                CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-          SPACE          *USRSPC
NAME-      OUTPUTBUFDF            TYPE-      19  SUBTYPE-      34
LIBRARY-   USRDFNCMN              TYPE-      04  SUBTYPE-      01
CREATION-  12/21/90 12:36:27      SIZE-      00000400
OWNER-     QSECOFR                TYPE-      08  SUBTYPE-      01
ATTRIBUTES-          0800          ADDRESS-    009FFE00 0000
SPACE ATTRIBUTES-
000000 00000080 00000060 1934D6E4 E3D7E4E3 C2E4C6C4 40404040 40404040 40404040 * - OUTPUTBUFDF *
000020 40404040 40404040 E0000000 00000000 00000200 00800000 00000000 00000000 * \ *
000040 00000000 00000000 0005004D 42000400 00000000 00000000 00000000 00000000 * (a *
SPACE-
000000 02000100 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
000020 02000100 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
000040 02000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
000060 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
LINES 000080 TO 0001FF SAME AS ABOVE
POINTERS-
NONE
OIR DATA-
TEXT-
000000 D8D7C1C4 C5E5F0F0 F0F2D8E2 C5C3D6C6 D9404040 F0F0F6F6 F2F7 *QPADEV0002QSECOFR 006627 *
SERVICE-
000000 40404040 40404040 40404040 40404040 40404040 40F14040 40404040 40404040 * 1 *
000020 40404040 40404040 404040E5 F2D9F1D4 F0F0F9F0 F1F2F2F1 F1F2F3F6 F2F74040 * V2R1M00901221123627 *
000040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 * *
000060 40404040 40404040 40404040 40404040 40404040 40404040 00000000 00000000 * *
000080 00000000 00000000 00000000 00000000 00000000 00000000 00000000 40400000 00000000 * *
0000A0 00000000 00000000 * *
END OF DUMP
* * * * * E N D O F L I S T I N G * * * * *

```

### User Spaces for Receiving Data, Operation X'0001'

Two user spaces are shown below. The first is the input buffer and the second is the input buffer descriptor.

The user spaces below are filled in showing that 2 data units were received. The first data unit has the more data indicator turned on in the buffer descriptor for the data unit. This means that the X.25 more indicator was turned on in all the X.25 packets that this data unit contains. The second data unit does not have the more data indicator turned on, indicating that the last X.25 packet in the data unit had the X.25 more indicator turned off. The first and second data unit are considered to be logically contiguous to the application program.

**Note:** This link was enabled specifying a data unit size of 1024 bytes. The sending system sent the data in data unit sizes of 512 bytes and they were combined into the 1024 byte data unit size by the local system. The data unit size is not negotiated end-to-end, neither is the maximum amount of contiguous data or the automatic flow control. Because the values are important, each application should be aware of what the other application has specified for each value. Refer to Sending and Receiving Data Packets for more information.

Figure 1-7. User Space (Buffer) Containing the Three Data Units



```

5738SS1 V2R1M0 910524      OS/400 DUMP      006625/QSECOFR/QPADEV0001  12/21/90 12:59:33  PAGE  1
DMPYSOBY PARAMETERS
OBJ- INBUFFER                CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-                SPACE                                *USRSPC
NAME-      INBUFFER          TYPE-      19  SUBTYPE-      34
LIBRARY-   USRDFNCMN        TYPE-      04  SUBTYPE-      01
CREATION-  12/21/90 12:40:03 SIZE-      00002200
OWNER-     QSECOFR          TYPE-      08  SUBTYPE-      01
ATTRIBUTES-      0800      ADDRESS-     00A00400  0000
SPACE ATTRIBUTES-
000000  00000080 00000060 1934C9D5 C2E4C6C6  C5D94040 40404040 40404040 40404040 * - INBUFFER *
000020  40404040 40404040 E0000000 00000000  00002000 00800000 00000000 00000000 * \ *
000040  00000000 00000000 0005004D 42000400  00000000 00000000 00000000 00000000 * (a *
SPACE-
000000  F0F10000 00000000 00000000 00000000  00000000 00000000 00000000 00000000 *01 *
000020  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000 * *
LINES 000040 TO 0001FF SAME AS ABOVE
000200  F0F20000 00000000 00000000 00000000  00000000 00000000 00000000 00000000 *02 *
000220  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000 * *
LINES 000240 TO 0003FF SAME AS ABOVE
000400  F0F30000 00000000 00000000 00000000  00000000 00000000 00000000 00000000 *03 *
000420  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000 * *
LINES 000440 TO 001FFF SAME AS ABOVE
POINTERS-
NONE
OIR DATA-
TEXT-
000000  D8D7C1C4 C5E5F0F0 F0F1D8E2 C5C3D6C6  D9404040 F0F0F6F6 F2F5 *QPADEV0001QSECOFR 006625 *
SERVICE-
000000  40404040 40404040 40404040 40404040  40404040 40F14040 40404040 40404040 * 1 *
000020  40404040 40404040 404040E5 F2D9F1D4  F0F0F9F0 F1F2F2F1 F1F2F4F0 F0F34040 * V2R1M00901221124003 *
000040  40404040 40404040 40404040 40404040  40404040 40404040 40404040 40404040 * *
000060  40404040 40404040 40404040 40404040  40404040 40404040 00000000 00000000 * *
000080  00000000 00000000 00000000 00000000  00000000 00000000 40400000 00000000 * *
0000A0  00000000 00000000 *
END OF DUMP
* * * * * E N D O F L I S T I N G * * * * *

```

Figure 1-8. User Space (Descriptor Element) Describing the Three Data Units

```

5738SS1 V2R1M0 910524      OS/400 DUMP      006625/QSECOFR/QPADEV0001  12/21/90 12:59:41      PAGE 1
DMPYSOBY PARAMETERS
OBJ- INBUFFERD              CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-                SPACE                *USRSPC
NAME-      INBUFFERD        TYPE-      19  SUBTYPE-    34
LIBRARY-   USRDFNCMN        TYPE-      04  SUBTYPE-    01
CREATION-  12/21/90 12:40:03 SIZE-      00000400
OWNER-     QSECOFR          TYPE-      08  SUBTYPE-    01
ATTRIBUTES-      0800      ADDRESS-    00A00200 0000
SPACE ATTRIBUTES-
000000 00000080 00000060 1934C9D5 C2E4C6C6 C5D9C440 40404040 40404040 40404040 * - INBUFFERD *
000020 40404040 40404040 E0000000 00000000 00000200 00800000 00000000 00000000 * \ *
000040 00000000 00000000 0005004D 42000400 00000000 00000000 00000000 00000000 * (a *
SPACE-
000000 04000100 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
000020 02000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
000040 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
      LINES 000060 TO 0001FF SAME AS ABOVE
POINTERS-
NONE
OIR DATA-
TEXT-
000000 D8D7C1C4 C5E5F0F0 F0F1D8E2 C5C3D6C6 D9404040 F0F0F6F6 F2F5 *QPADEV0001QSECOFR 006625 *
SERVICE-
000000 40404040 40404040 40404040 40404040 40404040 40F14040 40404040 40404040 * 1 *
000020 40404040 40404040 404040E5 F2D9F1D4 F0F0F9F0 F1F2F2F1 F1F2F4F0 F0F34040 * V2R1M00901221124003 *
000040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 * *
000060 40404040 40404040 40404040 40404040 40404040 40404040 00000000 00000000 * *
000080 00000000 00000000 00000000 00000000 00000000 00000000 00000000 40400000 00000000 * *
0000A0 00000000 00000000 * *
END OF DUMP
      * * * * * E N D O F L I S T I N G * * * * *

```

### User Space to Clear a Connection or Call, Operation X'B100'

This user space was filled in to end an SVC connection or clear an incoming call. No facilities or clear user data are requested with this, but cause and diagnostic codes are specified (these are not ISO or SNA codes).

Figure 1-9. User Space to Send an SVC Clear

```

5738SS1 V2R1M0 910524          OS/400 DUMP          006625/QSECOFR/QPADEV0001    12/21/90 13:14:48    PAGE 1
DMPYSOBYJ PARAMETERS
OBJ- OUTBUFFER                  CONTEXT-USRDFNCMN
OBJTYPE- *USRSPC
OBJECT TYPE-          SPACE          *USRSPC
NAME-      OUTBUFFER          TYPE-      19  SUBTYPE-      34
LIBRARY-   USRDFNCMN         TYPE-      04  SUBTYPE-      01
CREATION-  12/21/90 12:40:03   SIZE-      00002200
OWNER-     QSECOFR           TYPE-      08  SUBTYPE-      01
ATTRIBUTES-          0800      ADDRESS-     00A00A00 0000
SPACE ATTRIBUTES-
000000 00000080 00000060 1934D6E4 E3C2E4C6 C6C5D940 40404040 40404040 40404040 * - OUTBUFFER *
000020 40404040 40404040 E0000000 00000000 00002000 00800000 00000000 00000000 * \ *
000040 00000000 00000000 0005004D 42000400 00000000 00000000 00000000 00000000 * (a *
SPACE-
000000 0000BEBE 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * XX *
000020 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 * *
      LINES 000040 TO 001FFF SAME AS ABOVE
POINTERS-
NONE
OIR DATA-
TEXT-
000000 D8D7C1C4 C5E5F0F0 F0F1D8E2 C5C3D6C6 D9404040 F0F0F6F6 F2F5 *QPADEV0001QSECOFR 006625 *
SERVICE-
000000 40404040 40404040 40404040 40404040 40404040 40F14040 40404040 40404040 * 1 *
000020 40404040 40404040 404040E5 F2D9F1D4 F0F0F9F0 F1F2F2F1 F1F2F4F0 F0F44040 * V2R1M00901221124004 *
000040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040 * *
000060 40404040 40404040 40404040 40404040 40404040 40404040 00000000 00000000 * *
000080 00000000 00000000 00000000 00000000 00000000 00000000 40400000 00000000 * *
0000A0 00000000 00000000 * *
END OF DUMP
* * * * * E N D O F L I S T I N G * * * * *

```

## Error Codes

The system and user-defined communications support reports important information that is useful for determining recovery actions when an error occurs. This information is referred to as error codes that are reported either to the job log or to the QSYSOPR message queue. For a complete list of the messages that are signaled by the user-defined communications APIs, see "Messages" on page 283.

In some cases error codes are reported to your application in the error specific parameter. The following sections list the valid error codes. Some of the error codes represent actual coding errors, others only report additional information. For information about the error codes for the individual user-defined communications APIs, see User-Defined Communications Support APIs.

### Local Area Network (LAN) Error Codes

Figure 1-10 (page 299) shows the valid hexadecimal codes your application can receive as a result of a call to the QOLSEND API using operation code X'0000'. The codes indicate that the data was never sent on the line. Associated with these error codes is a message in QSYSOPR, indicating the device description that caused the error, and the error code. After receiving the error code, the link will still be enabled and usable.

These error codes indicate to your application that a coding error was made and should be corrected.

**Figure 1-10. Error Codes Received While Sending Data over LAN**

Error Code	Description	Cause
3300 2A55	Routing length not valid	Routing length is not valid, or length does not equal length in routing field.
3300 2A5D	Maximum frame size limit exceeded	Length of data is greater than maximum frame size supported by the source SAP
5300 2A7B	Access Control not valid	Access Control specified is not supported

Error Code	Description	Cause
3300 2AA9	SAP address not valid	SAP address is not configured in the line description
3300 2AA9	SAP address not valid	SAP address is not configured in the line description
3300 2AD4	Data length too small (Ethernet Version 2 only)	Data must be at least 48 bytes long (46 bytes of data, plus 2 bytes for the Ethernet type field)
3300 2AD5	Ethernet type field is not valid (Ethernet Version 2 only)	Ethernet type field (first two bytes of data)

## X.25 Error Codes

Figure 1-11 (page 300) shows the valid error codes your application can receive as a result of

- A call to the QOLSEND API with operation X'B400' to accept an SVC call
- A call to the Receive Data (QOLRECV) API which returns the results of the open connection request operation, X'B101'
- The connection failure indication, reported by operation X'B301'

These error codes indicate to your application that a coding error was made, or a failure condition occurred.

**Figure 1-11. Error Codes Reported on X'B001', X'B301', and X'B400' Operations**

Error Code	Description	Cause
1200 3122	Outgoing channel not available	The logical channel is still active and in the process of being deactivated
3200 3050	Restart in progress	Temporary condition; retry operation
3200 3172	Outgoing channel not available	Temporary condition; retry operation
3200 3368	Remote address length not valid	Remote address length not supported by the network
3200 3384	Facility field error	A facility was encoded incorrectly or a duplicate facility was encoded
3200 3388	Facility field too long	The total length of the facilities, which includes user-specified facilities, the NUI facility from the line description, and system generated facilities, exceeded X.25 limits (109 bytes)
3200 338C	Response restricted by fast select	User data is not allowed with restriction
3200 3394	User data not allowed	User data is not allowed on the call accept if fast select was not requested.
3200 33CC	Call user data length not valid	The length of call user data is greater than 16 and fast select is not selected.
4200 3210	Reset request transmitted	The virtual circuit was reset by the local system. Refer to cause and diagnostic codes to determine recovery.
4200 3220	Clear request transmitted	The virtual circuit was cleared by the local system. Refer to cause and diagnostic codes to determine recovery.
4200 3222	Clear request transmitted	The virtual circuit was cleared by the local system because there was a problem with the packet size in the call accept. This is either a configuration problem or a network problem. Verify that the default packet size in the line description is correct.

Error Code	Description	Cause
4200 3224	Clear request transmitted	The virtual circuit was cleared by the local system because there was a problem with the window size in the call accept. This is either a configuration problem or a network problem. Verify that the default window size in the line description is correct.
4200 3230	Restart request transmitted	The virtual circuit was cleared by the local system. Refer to cause and diagnostic codes for more information.
4200 3280	Time-out on call	Call timed out
4600 3134	Clear indication was received	The virtual circuit was cleared by either the remote system or the network. Refer to cause and diagnostic codes for more information.
4600 3138	Restart indication received	Temporary condition; refer to the cause and diagnostic codes reported to correct the problem, then retry the operation

Figure 1-12 (page 301) shows the valid error codes your application can receive as a result of a call to the QOLRECV API with an operation code returned as X'B101'.

These error codes indicate to your application that the connection was cleared or reset for the following reasons.

**Figure 1-12. Error Codes Reported on the X'B101' Operation**

Error Code	Description	Cause
3200 3388	Facility field too long	The total length of the facilities, which includes user-specified facilities, the NUI facility from the line description, and system generated facilities, exceeded X.25 limits (109 bytes)
3200 3394	User data not allowed	User data is not allowed when fast select is not selected.
3200 33CC	Call user data length not valid	The length of call user data is greater than 16 and fast select is not selected.
4200 3240	Time-out on reset	The clear request resulted in an X.25 reset, which timed out
4200 3284	Time-out on clear	The remote system did not respond to the CLEAR within the time-out value
4600 3134	Clear indication was received	The virtual circuit was cleared by either the remote system or the network. Refer to cause and diagnostic codes for more information.

Figure 1-13 (page 301) shows the valid error codes your application can receive as a result of a call to the QOLRECV API, returning the operation code, X'BF01'.

These error codes indicate to your application that the connection was cleared or reset for the following reasons.

**Figure 1-13. Error Codes Reported on the X'BF01' Operation**

Error Code	Description	Cause
3200 3050	Network Restart in progress	Temporary condition; connection is no longer active.
3200 3A0C	Close pending	The virtual circuit is being closed.

Error Code	Description	Cause
3200 3A0D	Reset pending	The virtual circuit is in the process of being reset by either the remote system or the network.
4200 3210	Reset packet transmitted	A Reset packet was transmitted from the local system.
4200 3240	Time-out on reset	The clear request resulted in an X.25 reset, which timed out
4600 3130	Reset indication was received	The virtual circuit received a reset by either the remote system or the network. Refer to cause and diagnostic codes for more information.
4600 3134	Clear indication was received	The virtual circuit was cleared by either the remote system or the network. Refer to cause and diagnostic codes for more information.

Figure 1-14 (page 302) shows the valid error codes your application can receive as a result of a call to the QOLSEND API with an operation code returned as X'0000'.

These error codes indicate to your application that the connection was cleared or reset for the following reasons.

**Figure 1-14. Error Codes Resulting from a X'0000' Operation**

Error Code	Description	Cause
3200 3050	Network restart in progress	Temporary condition; connection is no longer active.
3200 336A	Q/M bit sequence not valid	If the data is qualified, the Q bit must be set for all data units.
3200 33C8	Data length not valid	The length of the packet is not supported for this virtual circuit.
3200 3A0C	Close pending	The virtual circuit is being closed.
3200 3A0D	Reset pending	The virtual circuit is in the process of being reset by either the remote system or the network.
4200 3284	Interrupt timed out	The local DTE sent an interrupt packet. The response to this packet was not received within the time-out period, and the connection has been reset by the iSeries server.
4600 3130	Reset indication was received	The virtual circuit received a reset by either the remote system or the network. Refer to cause and diagnostic codes for more information.
4600 3134	Clear indication was received	The virtual circuit was cleared by either the remote system or the network. Refer to cause and diagnostic codes for more information.

## Common Errors and Messages

This section shows some of the common errors that you or your application programmer may encounter. Some of these errors are detected by the APIs and reported to the application by the unsuccessful return and reason codes returned on each API. Other errors are program design errors, that your application programmer must detect and correct. The errors are listed by category:

### Parameter Errors

- Switching use of connection identifiers (PCEP and UCEP)
- Switching use of timer handles
- Not encoding parameters if not used

- Operation code not in hexadecimal format
- Parameter not declared with proper length

### **User Space Errors**

- Not encoding reserved space for fields not used
- Not initializing user space fields as necessary.

The output user spaces can only be changed by the user-defined communications application. Operations are validated on each request. If there are fields that the current operation does not use, they should be set to contain zeros with X'00', to prevent a template error resulting from information on the previous operation still being in the user space. Not resetting the indicators in the output buffer descriptors on each operation and not zeroing out fields before making a call request may result in template errors.

### **Queue Errors**

- Queue not created
- Queue created with different key length than specified in the parameter list of the Enable Link (QOLELINK) API

### **Receive Data (QOLRECV) API Errors**

- Not checking the more data output parameter and issuing another call to the QOLRECV API
- Not calling the QOLSETF API to set the filter to route inbound data to the application
- Using the wrong data unit descriptor for the data unit (each data unit has its own descriptor)

### **Send Data (QOLSEND) API Errors**

- After a call to the QOLSEND API with an operation code of X'B000', X'B100', or X'BF00', the application should then call the QRCVDTAQ API and wait for incoming data to be placed on the queues. The success or failure of these operations is reported through the QOLRECV API with operation codes of X'B001', X'B101' and X'BF01', respectively.
- Using the wrong data unit descriptor for the data unit (each data unit has its own descriptor)

### **Enable Link (QOLELINK) API Errors**

- User space names not unique
- Queue not created before program call
- Line description not created or incorrect prior to program call

### **Query Line Description (QOLQLIND) API Errors**

- Parameter buffer not large enough

[Top](#) | ["Communications APIs," on page 1](#) | [APIs by category](#)





---

## Appendix. Notices

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

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

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

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

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

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

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
Software Interoperability Coordinator, Department YBWA  
3605 Highway 52 N  
Rochester, MN 55901  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

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

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

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

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

#### COPYRIGHT LICENSE:

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

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

---

## 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  
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 for downloading and printing publications

Permissions for the use of the information you have selected for download are granted subject to the following terms and conditions and your indication of acceptance thereof.

**Personal Use:** You may reproduce this information for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of this information, or any portion thereof, without the express consent of IBM<sup>(R)</sup>.

**Commercial Use:** You may reproduce, distribute and display this information solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of this information, or reproduce, distribute or display this information 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 information or any 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 information 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 THIS INFORMATION. THE INFORMATION IS 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.

All material copyrighted by IBM Corporation.

By downloading or printing information from this site, you have indicated your agreement with these terms and conditions.

---

## Code disclaimer information

This document contains programming examples.

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

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

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

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





Printed in USA