Using TCP and UDP Services

This chapter describes API considerations when using TCP or UDP as the underlying transport provider. It includes these sections:

- Protocol Address

  Describes the protocol address structure, transport layer address/port numbers, and network layer addressing/IP address.

- Expedited Data

  Describes how Cisco IOS for S/390 maps TCP “urgent data” into API expedited data, including sending and receiving expedited data.

- Disconnect Reason Codes

  Lists the disconnect reason codes generated when a disconnect indicated is presented to the transport user application program.

- API-initiated Protocol Actions

  Describes the actions performed by the TCP transport provider on behalf of the respective transport user API requests.

- Protocol Events Resulting In API Activity

  Describes the protocol events that result in API activity.

Protocol Address

The protocol address structure used by the API consists of a domain type followed by a protocol-dependent protocol address. This structure is mapped by the macro APIDTPA for the assembler API and the tpainet structure within the header api.h for the basic C library API.

The domain type portion of the protocol address used for TCP and UDP is INET (for Internet). The protocol-dependent portion of the INET domain type is the concatenation of the 16-bit TCP or UDP port number and the 32-bit IP network address.

Protocol dependencies apply to the structure of the Internet protocol address, the use of options, the use of API parameters, protocol dependent disconnect reason codes, and the definition of expedited data. A summary of how protocol events and actions relate to API events and actions is also included.

Transport Layer Address – Port Numbers

In most applications using the client/server model, port numbers for server programs are defined by the application, and port numbers for client programs are determined by, or requested from, the transport provider prior to the sending of datagrams or request for connection. Server programs typically use what are referred to as "well-known ports." A well-known port is a specific port number that is always the same no matter when, where, or how the server program executes. This mechanism lets client programs reliably request connections to the desired server application independent of other factors. Use of well-known ports by the server and dynamic assignment of client port numbers also lend themselves to the one-to-many relationship that client/server applications use.

When selecting a port number for a server application, developers must have knowledge of the well-known port numbers used by other applications on their networks as well as "official" port numbers that are reserved for specific applications on an Internet-wide basis. Official well-known port numbers are used by common applications such as File Transfer Protocol (FTP) and Virtual Terminal (TELNET). These port numbers are in the range of 1-255 and are listed in the *Assigned Numbers* RFC. If an application protocol that has an officially assigned port is being implemented, the respective port numbers should be specified in the application's address structures. Some existing applications use well-known port numbers that are unofficially assigned. Where application-to-port number mappings are defined is implementation-dependent (for example, file /etc/services on UNIX-based systems; SERVICE statements in APPCFGxx for Cisco IOS for S/390).

Like most implementations of TCP and UDP, Cisco IOS for S/390 divides the port number space between well-known (server) and client port numbers. In Cisco IOS for S/390, ports that can be used as server ports are defined by the PORTUSE keyword parameter on the TCP and UDP parameter statements. The default range for PORTUSE is 1-4095. Ports that are used for clients ports are specified on the PORTASGN keyword parameter of the TCP and UDP parameter statements. The default port range for PORTASGN is 4096-65535. Note that these are general conventions and exceptions are possible.

Local port numbers for TCP and UDP can be selected by the API application program by supplying the desired port number in the protocol address structure referenced by a TBIND request specifying OPTCD=USE. Local port numbers selected by the application must be less than 4096. Specifying local port numbers 4096 or greater would conflict with dynamic port assignment.

Applications request local port number assignment by issuing the TBIND request OPTCD=ASSIGN parameter. If an address buffer was supplied on the TBIND request, the transport provider returns the assigned local port number in the address structure.

An application that is requesting a TCP connection, UDP association, or UDP sending datagrams to a remote application must provide the remote port number that is used by that application. For TCP connections and UDP associations, this information is provided in the address structure referenced by the TCONNECT request. With UDP datagrams, the remote port is supplied with the TSENDTO request.

Server applications using the API normally do not need to know client port numbers. However, the transport provider passes the remote protocol address to the application on completion of a TLISTEN request, if an address buffer is provided. The remote protocol address is also available at any time after the connection or association has been established via the TADDR OPTCD=REMOTE function.

Network Layer Addressing – IP Address

Following the port number in the address structure is the IP network address of the host containing the specified port. The application program normally is not concerned with the local IP address. In the API, this information must be left as zero or truncated in the address structure when TBIND is issued. This is because it is possible for Cisco IOS for S/390 to have network interfaces to multiple networks and thus have multiple local IP addresses. It is the responsibility of the routing algorithms in Cisco IOS for S/390 to determine which local network interface and associated local IP address are used to transmit data. The selected local IP address is available to the application after the connection has been established by issuing a TADDR OPTCD=LOCAL API request.

A remote IP address must be provided when connecting or sending datagrams to a remote computer. The address structure supplied on a TCONNECT request must contain the remote IP address when establishing a TCP connection or UDP association. When sending UDP datagrams without the use of a UDP association, an address structure containing the remote IP address is supplied with each datagram on a TSENDTO request.

The API applications operating as servers do not normally require the remote IP address. However, it may be obtained by providing an address structure with the TLISTEN request. It may also be retrieved while the TCP connection or UDP association is in effect with a TADDR OPTCD=REMOTE API request.

An application that is using UDP without associations must supply an address buffer on the TRECVFR request if the remote source IP address is desired.

Expedited Data

Cisco IOS for S/390 maps TCP “urgent data” into API expedited data. There is no corresponding mechanism in the UDP protocol and therefore expedited data is not supported by the API when using UDP.

Sending Expedited Data

When an application issues a TSEND OPTCD=EXPEDITE request, the transport provider sets the TCP header urgent flag and sets the TCP header urgent pointer to the TCP sequence number offset corresponding to the TCP sequence number of the data byte following the last byte of TSEND OPTCD=EXPEDITE data. These are set in each TCP segment until all of the expedited/urgent data is transmitted.

Receiving Expedited Data

When urgent TCP data arrives from the network, Cisco IOS for S/390 notifies the transport user of expedited data. This is accomplished by setting the EXPEDITE bit in the TPL of any outstanding TRECV request. If there are no outstanding TRECV requests when urgent TCP data arrives, and the transport user has enabled an expedited data indication exit routine, the transport provider drives that exit. In either case, all subsequent TRECV requests complete with the EXPEDITE bit set until all of the urgent TCP data has been received by the transport user. The expedited data indication exit is also disabled until all of the expedited data has been received by the transport user.

A TRECV request that receives all remaining TCP urgent data receives only TCP urgent data, even if there is space for additional data in the receive buffer and more non-urgent TCP data has arrived at the transport provider.

Disconnect Reason Codes

When a disconnect indication is presented to the transport user application program, the program can determine the reason for the disconnect indication (and clear the indication) by issuing a TCLEAR request.

Symbolic Representations and Descriptions

The symbolic representations and descriptions of the disconnect reason codes are listed in this table. The symbolic disconnect reason codes are mapped into numeric codes by the APIDTPL macro with the DOMAIN=INET parameter:

TDTRANTO	Excessive unacknowledged retransmissions of TCP data caused the local transport provider to consider the TCP connection disconnected. This condition is sometimes referred to as retransmission time-out.
TDHOSTUN	An ICMP “host unreachable” message was received. This may be due to a routing configuration problem or a failure of some network component necessary to reach the desired destination.
TDPORTUN	An ICMP “port unreachable” message was received from the remote host. The desired port is either inactive or unsupported.
TDRABORT	A TCP segment was received with the RST (“reset”) bit on in the TCP header. This may be the result of the remote TCP detecting a fatal error in the TCP connection or may have been initiated by the application that is using TCP.
TDLNIDWN	The local network interface that is necessary to reach the remote host is inactive.
TDPROTUN	An ICMP “protocol unreachable” message was received from the remote host. TCP is inactive or unsupported on the desired destination.
TDACPRR	The connection has been terminated due to an unrecoverable error within some component of the ACP.
TDAPIRR	The connection has been terminated due to an unrecoverable error within some component of the API.

TDNETUN	An ICMP “net unreachable” message was received.
TDNOFRAG	An ICMP “fragmentation needed and DF set” message was received.
TDSRFAIL	An ICMP “source route failed” message was received.

API-initiated Protocol Actions

This section briefly describes the actions performed by the TCP transport provider on behalf of the respective transport user API requests. Only the API requests that cause some TCP-defined protocol activity are listed; API requests that are processed by the transport provider but do not direct any action by TCP are omitted. Also omitted are the protocol actions of UDP, because they reduce to the sending of data and, in the case of UDP associations, simulation of connection establishment and termination.

Macros and Associated Protocol Actions

This table lists the macros and the associated protocol actions:

Macro	Protocol Action
TCONNECT	Send initial SYN.
TDISCONN	Send TCP RST.
TRECV	Increase the available receive window space.
TREJECT	Send TCP RST.
TRELEASE	Send TCP FIN.
TSEND	Make data available for packetization; cause immediate transmission with PUSH bit set if OPTCD=NOMORE was in effect. TSEND requests are completed when all of the data sent is acknowledged by the remote TCP.

Protocol Events Resulting In API Activity

This section lists the protocol events that result in API activity.

Initial SYN Arrives (TCP)

If there is a TLISTEN pending completion within the provider, it is completed. If there was no pending TLISTEN, but the transport user had provided a connect indication exit, the exit is driven unless the exit had been previously driven and the transport user had not issued TLISTENs for all of the outstanding connect indications.

SYN/ACK Arrives in Response to a Previously Sent Initial SYN(TCP)

If there is a TCONFIRM pending completion within the provider, it is completed. If there was no pending TCONFIRM, but the transport user had provided a connect confirmation exit, the exit is driven.

Data Arrives (TCP/UDP)

If there are any pending TRECVs or TRECVFRs awaiting the arrival of data, completion occurs until there is no more data or no more requests. If there were no pending TRECVs or TRECVFRs, and the transport user had provided a data indication exit, that exit is driven. If UDP associations are in use, the first datagram to arrive is also considered a connect indication and is processed in the same manner as the arrival of an initial TCP SYN.

Acknowledgment for Sent Data Arrives (TCP)

The associated pending TSEND requests are completed. If the endpoint is operating in socket mode, the Send Window Open event is raised if the window was previously closed and there are no pending TSEND requests.

Urgent Data Arrives (TCP)

If there are no pending TRECVs awaiting data and the transport user had provided an expedited data indication exit, it is driven. The endpoint enters urgent mode. All TRECV requests complete with the EXPEDITE bit set in the TPL while in urgent mode.

ICMP Message Arrives

Any ICMP message that indicates a permanent error causes a disconnect indication to be generated. The disconnect indication is presented to the transport user via the completion of pending requests, if any. If not, the disconnect indication exit is driven, if it exists. Any subsequent connection termination request (except TCLEAR unless OPTCD=CLEAR was in effect) or data transfer request completes abnormally until the disconnect indication is cleared. In any case, the respective disconnect reason code is set within the endpoint and is available to the transport user via TCLEAR.

A TCP RESET Arrives

A disconnect indication is passed to the transport user and processed according to the same method that is used when a permanent ICMP error occurs.

A FIN Arrives (TCP)

If there is a TRELACK pending at the endpoint, it can be completed after all outstanding TRECVs have been completed. Any outstanding or subsequently issued TRECV for which there is no data is completed with a release indication in the return code. If the release indication is not presented in a TRECV request and the transport user had provided a release indication exit, the exit is driven after all TRECVs that completes normally (in other words, with data) have completed.