



- ▶ TCI/IP Support
- ▶ APL as a Server

APL as a TCP/IP Server

A Stream based APL server initiates a connection by creating a TCPSocket object whose SocketType is `'Stream'` (the default).

The service is uniquely identified on the network by the server's IP Address and port number, which are specified by the LocalAddr and LocalPort properties respectively. Note that unless you have more than one network adapter in your computer, LocalAddr is normally allowed to default.

This TCPSocket object effectively defines the availability of a particular service and at this stage is known as a *listening socket* which is simply waiting for a client to connect. This is reflected by the value of its CurrentState property which is `'Listening'`.

For example:

```

SO' WC' TCPSocket' ('LocalPort' 2001)
SO' WG' CurrentState'
Listening

```

When a client connects to the APL server, the state of the TCPSocket object (which is reported by the CurrentState property) changes from `'Listening'` to `'Connected'` and it generates a TCPAccept event. Note that the connection cannot be nullified by the return value of a callback function attached to this event.

At this point, you can identify the client by the value of the RemoteAddr property of the TCPSocket object. If you wish to reject a particular client, you must immediately expunge the (connected) TCPSocket and then create a new one ready for another client.

If you want to serve multiple clients, you must attach a callback function to the TCPAccept event and the callback must generate a new listening TCPSocket object as each client connects.

Serving Multiple Clients

Dyalog provides a special mechanism to enable a single server to connect to multiple clients. This mechanism is designed to accommodate the underlying operation of the Windows socket interface in the most convenient manner for the APL programmer.

What actually happens when a client connects to your server, is that Windows automatically creates a new socket, leaving the original server socket intact, and still listening. At this stage, APL has a single name (the name of your TCPSocket object) but two sockets to deal with.

Dyalog Ltd

South Barn
Minchens Court
Minchens Lane
Bramley
Hampshire
RG26 5BH
United Kingdom

Phone:

+ 44 (0) 1256 830 030

Fax:

+ 44 (0) 1256 830 031

e-mail:

sales@dyalog.com



▶ TCI/IP Support

▶ APL as a Server

As it would be inappropriate for APL itself to assign a new name to the new socket, it disassociates the TCPSocket object from its original socket handle, and reassociates it with the new socket handle. This is reflected by a corresponding change in its SocketNumber property. The original listening socket is left, temporarily, in a state where it is not associated with the name of any APL object.

Having performed these operations, APL checks to see if you have attached a callback function to the TCPAccept event. If not, APL simply closes the original listening socket. This then satisfies the simple case where the server is intended to connect with only a single client and your socket has simply changed its state from *'Listening'* to *'Connected'*.

If there *is* a callback function attached to the TCPAccept event, APL invokes it and passes it the window handle of the listening socket. What the callback must do is to create a new TCPSocket object associated with this handle. If the callback exits without doing this, APL closes the original listening socket thereby preventing further clients from connecting. If you wish to serve multiple clients, you must continually allocate new TCPSocket objects to the listening socket in this way so that there is always one available for connection.

The following example illustrates how this is done. Note that when the callback creates the new TCPSocket object, you **must not** specify any other property except SocketNumber, Event and Data in the `⎕WC` statement that you use to create it. This is important as the objective is to associate your new TCPSocket object with the original listening socket whose IP address and port number must remain unaltered.

Example: The original listening socket is created with the name *S0* and with a callback function *ACCEPT* attached to the TCPAccept event. The *COUNT* variable is initialised to 0. This variable will be incremented and used to generate new names for new TCPSocket objects as each client connects.

```

COUNT←0
'S0'⎕WC'TCPSocket' ('LocalPort' 2001) ('Event' 'TCPAccept' 'ACCEPT')
    
```

Then, each time a client connects, the *ACCEPT* function clones the original listening socket with a sequence of new TCPSocket objects using the name *S1*, *S2*, and so forth.

▽ *ACCEPT MSG*

```
[1]    COUNT←+1
```

```
[2]    ('S', ⌈COUNT)⎕WC'TCPSocket' ('SocketNumber' (3>MSG))
```

▽

Dyalog Ltd

South Barn
Minchens Court
Minchens Lane
Bramley
Hampshire
RG26 5BH
United Kingdom

Phone:

+ 44 (0) 1256 830 030

Fax:

+ 44 (0) 1256 830 031

e-mail:

sales@dyalog.com