Entire Net-Work Administration

Administration

Version 6.2.2

March 2013

Entire Net-Work

This document applies to Entire Net-Work Administration Version 6.2.2.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: WCPMF-ADMIN-622-20130315

Table of Contents

Entire Net-Work Administration

This document addresses administrators responsible for configuring and running an Entire Net-Work environment once the product is installed.

The Entire Net-Work Administration document is organized as follows:

TCP/IP Overview	Provides general information about TCP/IP and its use by Entire Net-Work Administration.
Simple Connection Line Driver Overview	Provides information for administrators responsible for configuring and running the Entire Net-Work Simple Connection Line Driver once Entire Net-Work Administration is installed.
Connecting to UES-Enabled Adabas Databases	Describes how to connect to UES-enabled databases.
Estimating Entire Net-Work Storage Requirements	Provides tables to assist in estimating the storage requirements of Entire Net-Work Administration.

1 Conventions

Notation *vrs* or *vr*: When used in this documentation, the notation *vrs* or *vr* stands for the relevant version, release, and system maintenance level numbers. For further information on product versions, see *version* in the *Glossary*.

This document covers the following topics:

Syntax Conventions

The following table describes the conventions used in syntax diagrams of Entire Net-Work statements.

Convention	Description	Example
uppercase, bold	Syntax elements appearing in uppercase and bold font are keywords. When specified, these keywords must be entered exactly as shown.	**DRIVER TCPI [DRVCHAR =** *driver-char* **| #]** The syntax elements DRIVER, TCPI, and DRVCHAR are Entire Net-Work keywords.
lowercase, italic, normal font	Syntax elements appearing in lowercase and normal, italic font identify items that you must supply.	**DRIVER TCPI [DRVCHAR =** *driver-char* **| #]** The syntax element *driver-char* identifies and describes the kind of value you must supply. In this instance, you must supply the special character used to designate that an operator command is directed to the TCP/IP line driver, rather than to a specific link.
underlining	Underlining is used for two purposes: 1. To identify default values, wherever appropriate. Otherwise, the defaults are explained in the accompanying parameter descriptions. 2. To identify the short form of a keyword.	**DRIVER TCPI [DRVCHAR =** *driver-char* **| #]** In the example above, # is the default that will be used for the DRVCHAR parameter if no other record buffer length is specified. Also in the example above, the short version of the DRVCHAR parameter is D.

Convention	Description	Example
vertical bars (|)	Vertical bars are used to separate mutually exclusive choices. **Note:** In more complex syntax involving the use of large brackets or braces, mutually exclusive choices are stacked instead.	DRIVER TCPI API = { IBM | HPS | BS2 | CMS | CNS | FUJ | ILK | OES } In the example above, you must select IBM, HPS, BS2, CMS, CNS, FUJ, ILK, or OES for the API parameter. There are no defaults.
brackets ([ ])	Brackets are used to identify optional elements. When multiple elements are stacked or separated by vertical bars within brackets, only one of the elements may be supplied.	DRIVER TCPI [DRVCHAR = *driver-char* | #] In this example, the DRVCHAR parameter is optional.
braces ({ })	Braces are used to identify required elements. When multiple elements are stacked or separated by vertical bars within braces, one and only one of the elements must be supplied.	DRIVER TCPI API = { IBM | HPS | BS2 | CMS | CNS | FUJ | ILK | OES } In this example, one of the following values is required for the API parameter: IBM, HPS, BS2, CMS, CNS, FUJ, ILK, or OES.
other punctuation and symbols	All other punctuation and symbols must be entered exactly as shown.	LINK *linkname* TCPI [NETADDR = *n1.n2.n3.n4*] [,][-] In this example, the periods must be specified in the IP address. In addition, options must be separated by commas and dashes should be used as needed to indicate that parameter settings continue on the next line.

Syntax Rules

The following rules apply when specifying Entire Net-Work parameter statements:

- Each Entire Net-Work parameter statement occupies positions 1 - 72 of at least one line.
- The statement type (NODE, LINK, or DRIVER) must be specified as the first nonblank item on the statement.
- The node name, driver name, translation definition function, or link name follows the statement type, separated by at least one blank (space).

- Keyword parameters may be specified following either the node name on NODE statements or the driver name on DRIVER and LINK statements. Keyword parameters are separated from their arguments by an equal (=) sign, and from other keyword parameters by at least one blank (space) or a comma (,).
- When the acceptable values for a parameter are Y and N (yes and no), any other value is treated as an N, unless there is a documented default, and processing continues without any warning.
- When the acceptable values for a parameter fall within a range (e.g., 1 - 2147483647) and a value outside the range is specified, the value is automatically reset to the maximum value within the range, unless documented otherwise for the parameter. Processing continues without any warning.
- A statement can be continued beginning in any column of the next line by specifying a dash (-) as the last nonblank character in any column of the current line, before column 73.
- Comment lines begin with an asterisk (*) in position 1 and can be inserted anywhere in the statement sequence.
- Some keywords may require a list of subparameters separated by commas; the list must be enclosed in parentheses ( ) unless only the first subparameter is to be entered. Omitted ("defaulted") subparameters must be represented by placeholder commas if subsequent parameters are to be entered. The following are examples of correct subparameter strings:

```
KEYWORD=(value1,value2,value3)
KEYWORD=(value1,,value3)
KEYWORD=(,,value3)
KEYWORD=(,value2)
KEYWORD=value1
```

- Hexadecimal keyword values can be entered by prefixing the value with an "X". For example:

```
LINK . . . ADJID=X0064, . . .
```

2 History of TCP/IP

In the late 1960s, the Defense Advanced Research Projects Agency (DARPA) of the United States Department of Defense initiated a project to interconnect computers. A small network was created that interconnected four computers, resulting in the creation of ARPANET. Colleges, universities, businesses, government offices, and military installations slowly began gaining access to ARPANET, and two large-scale communication backbone networks developed:

- the INTERNET for use by the government, colleges, and businesses; and
- the MILNET for exclusive military use.

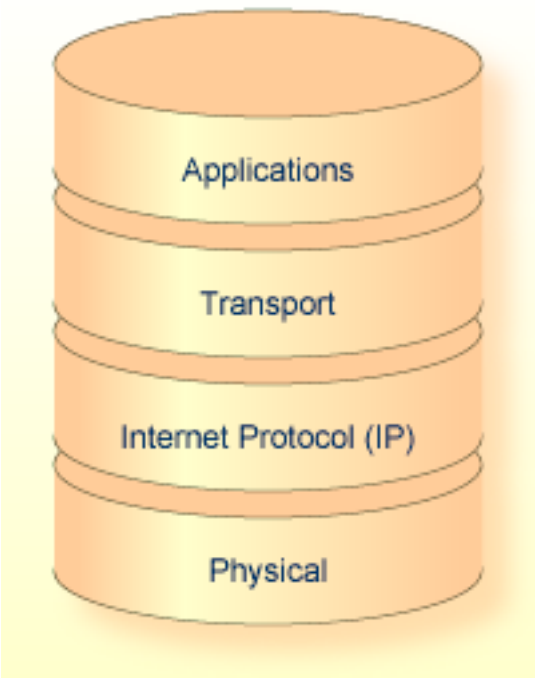
DARPA created a layered protocol stack in which each layer operates independently according to a set of prescribed rules known as Requests for Comments (RFCs) and Military Standards (MILSTDs). These are maintained and updated by the Department of Defense National Information Center (NIC) and the Internet Advisory Board (IAB).

Since its creation, the Transmission Control Protocol/Internet Protocol (TCP/IP) has been widely adopted as a standard in commercial applications, office automation, and personal computing networks. A copy of TCP/IP is integrated into the Berkeley UNIX Operating System, making TCP/IP the common protocol among UNIX systems.

3 TCP/IP Protocol Stack

As shown in the following diagram, the TCP/IP protocol stack contains four layers:

- Physical layer
- Internet Protocol (IP) layer
- Transport layer, comprising
  - Transmission Control Protocol (TCP); and
  - User Datagram Protocol (UDP)
- Applications layer



Physical Layer

At the bottom of the stack is the physical layer, which deals with the actual transmission of data over physical media such as serial lines, Ethernet, token rings, FDDI rings, and hyperchannels. Messages can also be sent and received over other, non-physical access methods such as VTAM/SNA.

Internet Protocol (IP) Layer

Above the physical layer is the Internet Protocol (IP) layer, which deals with the routing of packets from one computer to another. The IP layer

- determines which lower level protocol to use when multiple interfaces exist.
- determines whether to send a packet directly to the host or indirectly to a relay host known as a router.

  When a packet is larger than the size supported by the physical medium, the IP layer breaks the packet into smaller packets, a process referred to as "fragmentation and reassembly".
- provides some control services for packets, and ensures that they are not sent from router to router indefinitely.

However, the IP layer does not keep track of a packet after it is sent, nor does it guarantee that the packet will be delivered.

Transport Layer

Above the IP layer is the transport layer, which contains the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).

Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) guarantees that data sent by higher levels is delivered in order and without corruption. To accomplish this level of service, the TCP implementation on one computer establishes a session or connection with the TCP implementation on another computer. This process is referred to as Connection Oriented Transport Service (COTS).

After a session is established, data is sent and received as a stream of contiguous bytes; each byte can be referenced by an exact sequence number. When data is received by the remote TCP, it sends an acknowledgment back to the local TCP advising it of the sequence number of the last byte of data received. If an acknowledgment is not received, or if an acknowledgment for previously sent data is received twice, the local TCP retransmits the data until it is all acknowledged. The remote TCP discards any bytes that are received more than once.

All data sent and received by TCP is validated for corruption using checksums. Whenever a checksum is incorrect, the bad data is discarded by TCP, and the correct data is retransmitted until it is accurately received.

User Datagram Protocol (UDP)

Unlike the TCP, the User Datagram Protocol (UDP) transmits and receives data in packets (datagrams), and delivery is not guaranteed. The contents of the data can be sent with or without a checksum. The use of checksums varies widely from one implementation to another.

Applications Layer

Above the transport layer is the applications layer, which contains both general applications and function libraries for use by applications.

Some general applications that run over TCP include

- File Transfer Protocol (FTP);
- remote terminal emulation (TELNET in line mode, TN3270 in full screen);
- Electronic Mail (SMTP); and
- Entire Net-Work.

Some general applications that run over UDP are

- Network File Server (NFS); and
- Domain Name Server (DNS).

Interface with TCP and UDP

Function libraries provide routines to simplify the interface between applications and TCP/UDP of the Transport layer:

- The most common function library is known as Sockets, which allows an application written in C to access TCP as if it were just another stream input/output device.
- Another function library that is less commonly used is Remote Procedure Call (RPC), which allows applications to make calls to functions that are located in another application on a different computer.

The environment in which an application runs often dictates the interface used between it and TCP or UDP:

- Most UNIX, OS/2, and Windows applications are written in C and utilize a direct socket interface.
- On IBM mainframes and other systems based on the same architecture such as Fujitsu Technology Solutions, applications are often written in S/390 assembler, and use either a pseudo-socket interface or an application program interface (API) to gain access to the TCP/IP protocol stack.

Ports

The interface that exists between an application and TCP is referred to as a port. Ports are classified as server ports and client ports:

- Server ports are generally ports on which the application "listens" for incoming connections to be made.
- Client ports are generally ports on which the application "connects" outwardly to a server port.

  An application may control multiple client ports and server ports simultaneously.

  Each port is identified by a port number, which ranges from 1 to 65535.
- The port number used by client ports usually has no significance and is often assigned by TCP.
- Server port numbers, however, are usually required to be "well known"; that is, the client must know which port the server is listening on when it attempts to connect. Server port numbers usually are specified by the server application.

4 Supported TCP/IP Transport Providers

Under z/OS, IBM's eNetwork Communications Server is supported.

IBM TCP/IP Communication Subsystem and API

In z/OS environments, OpenEdition sockets (API=OES) or the high performance native sockets interface (API=HPS) are used to communicate with the IBM TCP/IP address space. A valid OpenEdition security context, referred to as an OMVS segment, must be defined for Entire Net-Work. For more information, see the IBM document z/OS OpenEdition Planning Guide.

If the OMVS segment is not properly defined, TCP/IP driver initialization will fail with error 156. The Entire Net-Work error message received is NETP600I if API=HPS or NETP700I if API=OES.

5 TCP/IP API Modules

The following table contains a list of available TCP/IP API modules. Some support the Domain Names Services GetHostByName and GetHostByAddr.

API	Interface Loaded	GetHostByName	GetHostByAddr
HPS	IBM interface NWTCPHPS (High Performance Native Sockets)	Yes	Yes
OES	IBM OpenEdition sockets interface NWTCPOES	Yes	Yes

6 Problem Determination

If you have questions or difficulties concerning the installation or operation of this product, contact your Software AG technical support representative. Before doing so, however, Software AG recommends that you have the following information available:

- The type and release level of the operating system being used.
- A brief description of the system configuration; for example, the types of and number of partners in the network, the software being used by the partners.
- A brief description of the problem you are experiencing.
- A hard copy of the Entire Net-Work DDPRINT DD card output file and the Entire Net-Work console log showing all Entire Net-Work write-to-operators (WTOs).

If Software AG support personnel request tracing information, they will indicate the necessary parameters and provide you with the appropriate settings for these parameters.

7 Simple Connection Line Driver Overview

The Entire Net-Work Simple Connection Line Driver provides communication between client applications that use Software AG's new e-business connections and Adabas databases running on z/OS, z/VSE, and BS2000 systems. The new e-business connections make use of:

- an enhanced communication protocol provided by Software AG that links e-business applications with enterprise servers
- the Software AG Directory Server (ADI).

Software AG products that support the e-business communication protocol and the Software AG Directory Server currently include Tamino, Jadabas, Entire Net-Work 7 and any other product that transports client requests using Software AG's ADALNKX module. The underlying transport mechanism is TCP/IP.

Note: The Simple Connection Line Driver cannot connect to another Simple Connection Line Driver.

The Software AG Directory Server is a centralized component that provides all directory information required to communicate between clients and servers and eliminates the need for directory configuration files on every machine. The code for the Software AG Directory Server is included in the Entire Net-Work Client code available on Software AG's Empower. To use the Simple Connection Line Driver, you must have the Software AG Directory Server installed somewhere on your system. If you have already installed this code with another Software AG product, we recommend that you use the installed code, so that your organization uses only one shared Directory Server. For more information about the Software AG Directory Server, read *Software AG Directory Server Documentation* in the *Software AG Directory Server Administration Guide*. The documentation for Entire Net-Work Client is included with its code on Empower.

When using the new Simple Connection Line Driver, mainframe Adabas conversion must be enabled for all the databases that will be called by requests coming through the driver. For information on enabling Adabas conversion, read ***Simple Connection Line Driver Prerequisites***, elsewhere in this guide.

This chapter provides information for administrators responsible for configuring and running the Entire Net-Work Simple Connection Line Driver once Entire Net-Work is installed.

The Simple Connection Line Driver documentation is organized as follows:

How the Entire Net-Work Simple Connection Line Driver Operates	Describes how the Simple Connection Line Driver works.
Simple Connection Line Driver Prerequisites	Lists prerequisites for installing and using the Simple Connection Line Driver.
Installing the Simple Connection Line Driver Under z/OS, z/VSE, or MSP	Describes the installation of the Simple Connection Line Driver on z/OS, z/VSE, or MSP host systems.
Installing the Simple Connection Line Driver Under BS2000/OSD	Describes the installation of the Simple Connection Line Driver on BS2000/OSD operating systems.
Connecting to Entire Net-Work 7	Describes Entire Net-Work 7 support provided by the Simple Connection Line Driver, its limitations, and what you need to do to implement it.
TCPX DRIVER Statement	Describes the syntax and parameters of the TCPX DRIVER statement.
TCPX LINK Statement	Describes the syntax and parameters of the TCPX LINK statement.
Simple Connection Line Driver Operator Commands	Describes the operator commands you can use with the Simple Connection Line Driver.
Model Links	Describes the model link support offered by the Simple Connection Line Driver.
TCPX Statistics	Describes the statistics provided by the Simple Connection Line Driver you can use to help tune each link and the driver itself.

8 How the Entire Net-Work Simple Connection Line Driver Operates

The Entire Net-Work Simple Connection Line Driver uses TCP/IP as its transport mechanism. The Simple Connection Line Driver establishes a TCP/IP listen on the port specified in the driver SERVERID parameter. This port matches the port specified in the access URL in the Directory Server used by the ADI-enabled client. (For complete information about Software AG Directory Server, read *Software AG Directory Server Documentation* in the *Software AG Directory Server Administration Guide*.)

When a client makes an Adabas call, the Directory Server URL information for the specified database ID is used to direct the call to the Simple Connection Line Driver. The Simple Connection Line Driver passes the call to the Entire Net-Work mainline, which makes the Adabas call.

The ADI-enabled client needs to know only the database ID. No client-side configuration or application changes are required. Client applications do not need to be modified; however, the Directory Server must be configured with the URL information for each target database.

The Directory Server does not participate in the data flow. It participates only in client-server session creation, by providing the client the correct network address of the target database server.

URLs contained in the Directory Server have the following format:

XTSaccess.*targetid***[0]=***protocol*://*host*:*port*[?*parm1*=*value*[&*parm2*=*value*]...]

For example:

```
XTSaccess.68[0]=TCPIP://ahost:3001?retry=3
```

In this example:

Entry	Description
XTSaccess	Identifies this as a Directory Server Access entry, read by the client.
68	The target database ID.
TCPIP	The communications protocol to be used for database 68.
ahost	The name of the host computer on which the server (Entire Net-Work with the Simple Connection Line Driver runs.
3001	The port where the server (the Simple Connection Line Driver) will listen. This value matches the `SERVERID` parameter of the TCPX DRIVER statement.
retry=3	One of multiple optional parameters that may be used. The first parameter is preceded by a question mark (?) and subsequent parameters, if any, are preceded by an ampersand (&).

Because all Software AG ADI-enabled products use the Software AG Directory Server, one may already be in place and configured for your network environment. Contact Software AG if you require additional assistance.

9 Simple Connection Line Driver Prerequisites

The following are prerequisites for using the Simple Connection Line Driver:

- The Software AG Directory Server (ADI) must be installed on the client application. ADI-enabled clients are clients that use the Software AG Directory Server. For complete information about Software AG Directory Server, read *Software AG Directory Server Documentation* in the *Software AG Directory Server Administration Guide*.
- The Adabas SVC must be at version 7.1.2 or later.
- Mainframe Adabas data conversion (Universal Encoding Support) services must be enabled for all databases.

  When using the new Simple Connection Line Driver, Adabas conversion must be enabled for all the databases that will be called by requests coming through the driver. The Simple Connection Line Driver does not provide any data conversion itself. Adabas 7 Universal Encoding Support (UES) performs any required data conversion, such as converting data for Adabas buffers between different machine architectures (ASCII, EBCDIC, Big Endian, Little Endian). For information on enabling Adabas Universal Encoding Support, read ***Connecting to UES-Enabled Adabas Databases*** elsewhere in this guide.

10 Installing the Simple Connection Line Driver Under z/OS, z/VSE, or MSP

This section describes the installation steps for installing Simple Connection Line Driver on a z/OS host systems.

Note: The Simple Connection Line Driver is not supported on MSP systems.

Installation Overview

To gain an understanding of the entire installation process, Software AG recommends that you read all of the installation steps before you perform the individual steps:

Step	Description
1	Verify the connectivity between cooperating nodes. This step is necessary only to support the TCP/IP line driver. It is not necessary for the Simple Connection Line Driver.
2	Unload the Entire Net-Work mainline and Entire Net-Work TCP/IP line driver libraries.
3	Outline your Entire Net-Work configuration and obtain all of the data necessary to configure the Entire Net-Work startup parameters.
4	Alter the Entire Net-Work startup job
5	Add the TCPX-specific DRIVER and LINK statements.
6	Configure client information. This step is necessary only to support the Simple Connection Line Driver. It is not necessary for the TCP/IP line driver.
7	Ensure that ADARUN has been linked with attribute AMODE(31). This step is necessary only to support the TCP/IP line driver. It is not necessary for the Simple Connection Line Driver.
8	Start Entire Net-Work and perform installation verifications.

Installation Steps

Step 1. Verify the connectivity between cooperating nodes .

Note: This step is necessary only to support the TCP/IP line driver. It is not necessary for the Simple Connection Line Driver.

Before installing the Entire Net-Work TCP/IP line driver, verify the connectivity between cooperating nodes. Use the PING utility from the remote system to perform a loopback test with the local IBM TCP/IP node as follows:

```
PING node-name
```

where node-name is the name that identifies the IBM node.

Step 2. Unload the Entire Net-Work mainline and Entire Net-Work TCP/IP Line Driver libraries.

If not already performed, install the Entire Net-Work mainline libraries, using the procedure for your operating system environment (z/OS). For more information, see the section *Installation Overview* in the *Entire Net-Work Installation Guide*. Then unload the Entire Net-Work TCP/IP line driver components from the installation tape as follows:

Under z/OS

Under z/OS, use IEBCOPY to restore the required data sets. Refer to the *Software AG Product Delivery Report* for the correct data set sequence numbers and names.

Under z/VSE

Under z/VSE, use LIBR RESTORE to restore the required data sets. Refer to the *Software AG Product Delivery Report* for the correct data set sequence numbers and names.

Under MSP

Under MSP, use MSP IEBCOPY to restore the required data sets. Refer to the *Software AG Product Delivery Report* for the correct data set sequence numbers and names.

Step 3. Outline your Entire Net-Work configuration and obtain all data necessary to configure the Entire Net-Work startup parameters.

Note: If you are only interested in using the Simple Connection Line Driver, this step is not necessary.

Before you set up your network parameters, obtain the following information:

For the Local IBM Node

- The name of the node;
- The number of the Adabas SVC to be used;
- The subsystem name of the TCP/IP transport provider;
- The job name or started task name of the TCP/IP transport provider;
- The well known port number (referred to on the IBM mainframe as the SERVERID parameter of the TCP/IP Driver statement) that Entire Net-Work will use to listen for incoming connections.

For Each Remote Node

Note: This is not necessary or recommended if you plan to use the Simple Connection Line Driver.

- The name of each node;

- The node's Internet Protocol (IP) address or, if a Domain Name Resolver is being used, the Internet host name;
- The well known port number (referred to on the IBM mainframe as the SERVERID parameter of the DRIVER/LINK statement) that Entire Net-Work will use when establishing a connection to that node.

Step 4. Alter the Entire Net-Work startup job.

Make the following changes to the Entire Net-Work startup JCL.

Under z/OS

A sample startup JCL member called JCLNET is provided in the source libraries. Then add the Simple Connection Line Driver load library to the STEPLIB concatenation.

Under z/VSE

Alter the Entire Net-Work startup job to add the Simple Connection Line Driver library/sublibrary to the LIBDEF search chain. (See the sample source member JCLNET in the source library for an example of Entire Net-Work startup JCL.)

Under MSP

Alter the Entire Net-Work startup job to add the Simple Connection Line Driver load library to the STEPLIB concatenation.

Step 5. Add the TCPX-specific DRIVER and LINK statements.

Use the Entire Net-Work configuration built in Step 3, along with the parameters of the DRIVER and (optionally) LINK statements, to create the necessary DRIVER and LINK statements for your environment in the Entire Net-Work DDKARTE input file.

Note: If you are installing the Simple Connection Line Driver on z/VSE systems, the value of the TCPX DRIVER's API parameter must be "CNS".

Step 6. Configure client information.

Note: This step is necessary only to support the Simple Connection Line Driver. It is not necessary for the TCP/IP line driver.

In order for a client to correctly send the database request to the Entire Net-Work node where the database is located, Software AG Directory Server entries must be added for each database. These entries tell the client application where the server (Entire Net-Work) is located and which databases it serves. A Directory Server access entry must be added for each database that the client will call via the Simple Connection Line Driver.

A Directory Server access entry looks like this:

XTSaccess.*targetid***[0]=***protocol***://***host***:***port***[?***parm1***=***value***[&***parm2***=***value***]...]**

where *targetid* is the target database ID, *protocol* is the communication protocol to use, *host* is the name of the host computer where the server (Entire Net-Work with the Simple Connection Line Driver) runs, *port* is the port on which the server (Simple Connection Line Driver) will listen, *parm1* and *parm2* are optional parameter names and *value* represents the values of those parameters. The port number must match the Simple Connection Line Driver's SERVERID parameter. If one Simple Connection Line Driver will serve multiple databases, an Access entry for each database is required, but these entries would all specify the same port number.

For example:

```
XTSaccess.68[0]=TCPIP://ahost:3001
```

In this example, the target database is database 68. It should be accessed using the TCP/IP protocol on the "ahost" computer at port 3001.

For more information about the Directory Server, read *Software AG Directory Server Documentation* in the *Software AG Directory Server Administration Guide*.

Step 7. Ensure that ADARUN has been linked with AMODE(31).

Note: This step is necessary only to support the TCP/IP line driver. It is not necessary for the Simple Connection Line Driver.

The TCP/IP driver attempts to allocate all private virtual storage above the 16-megabyte line. To take advantage of this feature, ensure that ADARUN has been linked with attribute AMODE(31).

Step 8. Start Entire Net-Work and verify the installation.

Because of the many possible variations of the Entire Net-Work, Adabas, and applications topology, Software AG does not provide standard installation verification procedures. However, the following procedure is suggested for verifying the TCP/IP line driver installation:

1. Start the Entire Net-Work system and make connections to each link defined to the system.
2. Test the connections and verify that the links can be established from either side by connecting and disconnecting the links several times from each node. While the links are connected, issue the Entire Net-Work operator command DISPLAY TARGET to display the targets and the nodes on which they are located.
3. Test your applications running across Entire Net-Work. At first, run one application at a time, then verify the results.

4. For the final verification test, run a load test through the network (that is, multiple users on each node accessing data on the partner node).

The following procedure is suggested for verifying the Simple Connection Line Driver installation:

1. Start the Entire Net-Work system.
2. Test your applications running across Entire Net-Work. At first, run one application at a time, then verify the results.
3. For the final verification test, run a load test through the network (that is, multiple users on each node accessing data on the partner node).

11 Installing the Simple Connection Line Driver Under BS2000/OSD

This section describes the procedure for installing Simple Connection Line Driver on BS2000/OSD host systems.

The information in this section uses the substitution variable *vrs*, which stands for the current version, revision, and system maintenance (SM) level of the product; for example: "612".

Installation Prerequisites

The Entire Net-Work TCP/IP line driver uses the Sockets subsystem which is a component of Open Net Server.

Installation Steps

1. Ensure that BCAM is up and running.
2. Ensure that the Sockets subsystem has been created and started. Check the status and version of the Sockets subsystem using the following command:

   ```
   SHOW-SUBSYSTEM-STATUS SUBSYSTEM-NAME=SOCKETS
   ```

3. Ensure that the appropriate entries have been added to the BCAM resource definitions (TCP partners, etc.) or that DNS name resolution is enabled.

Starting the Simple Connection Line Driver

The Simple Connection Line Driver consists of a component running in the main Entire Net-Work task and one separate subtask. The subtask is the actual TCP protocol handler. It performs the TCP/IP processing.

If you are running Sockets 2.2 or later, incoming calls to the Entire Net-Work from Sockets is signaled by a P1 event. A separate subtask is no longer necessary to maintain asynchronous processing.

If you are running a version of SOCKETs earlier than 2.2, two tasks are required for the following reasons:

- Both Adabas and the Sockets interface require a synchronous wait. Only one synchronous wait per task is possible.
- The BS2000/OSD kernel does not support multiple tasks within one address space. Socket calls are therefore processed by a separate operating system task, referred to as the TCP/IP line driver subtask.

The subtask is started automatically by the main task according to the parameters in the NWBS2SUB variable in the STARTP procedure; no separate operator action is required. The parameters for the subtask are constructed within the STARTP procedure. If necessary, however, the JOB-CLASS, LIBrary, or ELEment values may be changed.

```
/ DECLARE-VARIABLE -
/       NWBS2SUB(INIT=-
                     /'ENTER-PROC FROM  = (LIB = WCP&VERSION..LIB,-
                     / ELE = SUBTASKPROC),-
                     / PROC-PAR= (WTCLIB= WTC&VERSION..LIB,-
 /WCPLIB= WCP&VERSION..LIB,-
 /DUMP = &DUMP),-
 /JOB-CLAS = *STD,-
 / RESOURCE= *PAR(CPU-LIMIT=&TIME),-
 / SYS-OUTP = *DELETE)')
```

Caution: Never attempt to start the subtask manually!!

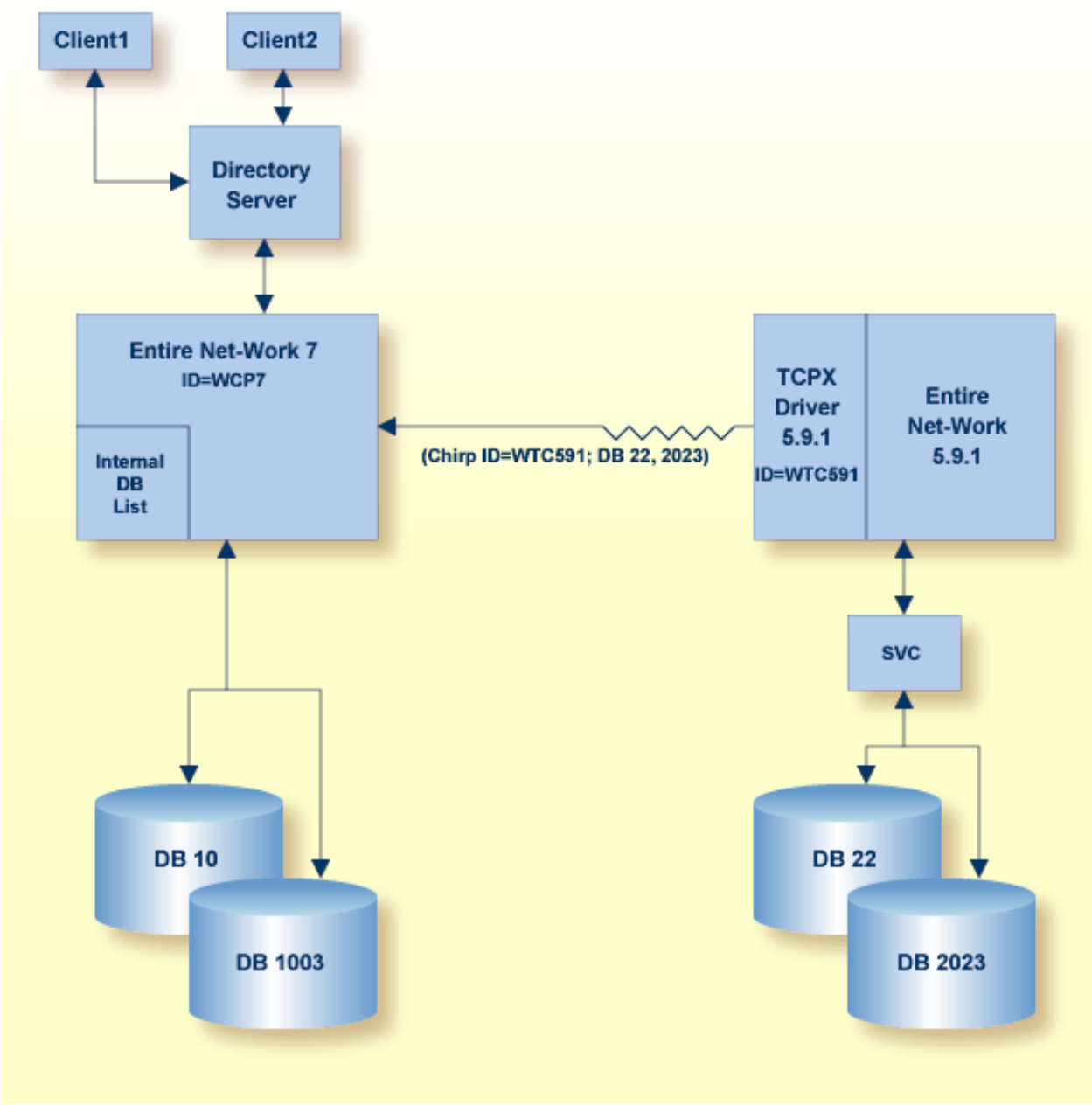
12 Connecting to Entire Net-Work 7

Using the Simple Connection Line Driver, you can connect your Entire Net-Work on open systems (Entire Net-Work 7 or later) to databases on the Simple Connection Line Driver node. This connection happens automatically when you start systems with Entire Net-Work 7 or later installed or Entire Net-Work systems with the Simple Connection Line Driver installed. The communications mechanism by which this connection occurs between Entire Net-Work 7 and the Simple Connection Line Driver is referred to as *chirping*. This should not be confused with the broadcast messages that Entire Net-Work sends about node availability. Broadcast messages are similar in function, but do not occur between Entire Net-Work 7 and the Simple Connection Line Driver and do not provide target-level availability information.

Note: If you do not have Entire Net-Work 7 installed, contact your Software AG technical support representative about acquiring it or about acquiring the Entire Net-Work Client available for download from Software AG's Empower web site (***https://empower.software-ag.com***).

When an Entire Net-Work 7 or Simple Connection Line Driver system is started, it chirps that it is available to its internal list of known Entire Net-Work connections. In addition to indicating that the system is now available, the chirp identifies which Adabas databases the system has access to and what their availability is. The system receiving the chirp can then establish and store, through its own processing, the URL of the system that chirped. The end result is that Entire Net-Work 7 systems and Entire Net-Work systems with Simple Connection Line Driver installed can readily access each other's databases.

The following diagram depicts the interaction between systems with the Simple Connection Line Driver installed and Entire Net-Work 7 systems.



There is one limitation to these connections.

- If you want point-to-point connections to the actual databases, a second Directory Server or a different partition in the existing Directory Server is needed. For more information, refer to your *Software AG Directory Server Documentation* in *Software AG Directory Server Administration*.

13 TCPX DRIVER Statement

The TCPX DRIVER statement and its parameters are used to activate and define the characteristics of the local IBM mainframe node. The access method name TCPX instructs Entire Net-Work to load the line driver module NETTCPX.

In most cases, only one DRIVER statement needs to be coded in your Entire Net-Work startup parameters. However, multiple DRIVER statements can be defined to allow Entire Net-Work to listen on multiple ports.

DRIVER Statement Format

The TCPX DRIVER statement has the following format:

```
DRIVER  TCPX  [ACCEPTUI = { N | Y } ]
              API = { HPS | OES }
              [APITRACE = (N | Y, N | Y, N | Y, N | Y, N | Y, N | Y, N | Y, N | Y, N | Y ) ]
              [CONNQUE = { n |  10  } ]
              [DRVCHAR = { character  |  # } ]
              [DRVNAME = { driver-name  |  TCPX } ]
              [KEEPALIV = { Y |  N } ]
              [MULTSESS = { N | Y  } ]
              [NUMUSERS = { number  |  100 } ]
              [OPTIONS1 = ( n, n, n, n, n, n, n, n, n, n ) ]
              [OPTIONS2 = ( x, x, x, x, x ) ]
              [PSTATS = { Y |  N } ]
              [RESTART = (interval, retries} ]
              [RSTATS = { Y |  N } ]
              [SERVERID = { n  |  1996} ]
              [STATINT = { stat-interval  |  3600 } ]
              [SUBSYS = { subsys-name  |  VMCF } ]
              [TRACE = { Y | N } ]
              [TRACELEV = (N | Y, N | Y, N | Y, N | Y, N | Y, N | Y, N | Y, N | Y, N | Y ) ]
              [TRACESIZ = { size  |  4096 } ]
              [USERID = { userid | TCPIP } ]
```

For more information about syntax conventions and rules used in this section, read *Conventions*.

Modifying the DRIVER Statement Parameters

The DRIVER statement parameters are read from a sequential file during system startup, and can be modified after startup using the ALTER operator command. Some parameters can be modified when the line driver is open or closed. Others can be modified only when the line driver is closed. Read about the ALTER and CLOSE commands in ***TCPX Operator Commands***. The open/closed requirement for each parameter is included in the parameter descriptions.

DRIVER Statement Parameters

This section describes all of the parameters that can be used for the TCPX DRIVER statement.

For more information about syntax conventions and rules used in this section, read ***Conventions***.

ACCEPTUI Parameter

```
ACCEPTUI = { Y | N }
```

This optional parameter determines whether the line driver will accept connections from systems that have not been previously defined with LINK statements. The ACCEPTUI parameter can be modified when the line driver is open or closed.

Valid values are "Y" (Yes) or "N" (No).

- If "Y" is specified, Entire Net-Work will accept connection requests from an undefined system and the required control blocks are built dynamically. Normal "handshaking" procedures with the new connections are performed. This is the default.
- If "N" is specified, Entire Net-Work will reject incoming requests from unknown source nodes.

API Parameter

```
API = { HPS | OES }
```

This required parameter specifies the name of the TCP/IP application program interface being used. The API parameter can be modified only when the line driver is closed. Supported values are shown in the table below. There is no default.

Value	Description	Valid for Platforms
HPS	Loads the IBM interface NWTCPHPS (High Performance Native Sockets)	z/OS and OS/390
OES	Loads the IBM OpenEdition sockets interface NWTCPOES	z/OS and OS/390

APITRACE Parameter

```
APITRACE= { N | Y, N | Y, N | Y, N | Y, N | Y, N | Y, N | Y, N | Y, N | Y, N | Y}
```

This optional parameter specifies debugging trace levels. It is a series of flags that are passed to the API routine, which then determines the events to be traced. The APITRACE specification must be enclosed in parentheses. For example:

```
APITRACE=(N,N,N,N,N,N,N,N,N,N)
```

Trace levels are positional within the parameter syntax-example and are set using Y (Yes) and N (No). It is recommended that all flags remain set to N, the default value. If your system is experiencing problems, contact your Software AG technical support representative for the settings that will produce the appropriate trace information. The APITRACE parameter can be modified when the line driver is open or closed.

CONNQUE Parameter

```
CONNQUE = {n | 10}
```

This optional parameter specifies the number of connect queue entries. The value specified must accommodate the maximum number of simultaneous connection requests from remote nodes.

After the connection is accepted or rejected, connect queue entries are reused. If the value of this parameter is not high enough, the API routine is not able to process the incoming connection and the partner eventually will time out. Depending on the API being used, a message may be displayed indicating that an error has occurred. Values can range from 1 to 64; a value greater than 64 is reset to 64. The default value is 10. The CONNQUE parameter can be modified only when the line driver is closed.

DRVCHAR Parameter

```
DRVCHAR = {char | # }
```

This optional parameter specifies the special character used to designate that an operator command should be directed to this line driver rather than to a specific link. The DRVCHAR parameter can be modified only when the line driver is closed.

The default for this parameter is "#".

DRVNAME Parameter

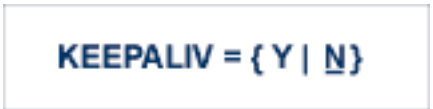
```
DRVNAME = {name | TCPX}
```

This optional parameter specifies the 4-byte driver name. The DRVNAME parameter can be modified only when the line driver is closed.

The default for this parameter is "TCPX".

The DRVNAME parameter enables sites to make multiple TCP/IP API routines available at the same time. For example, the IBM APIs can be made available within the same Entire Net-Work address space. This parameter also allows two or more drivers to be defined so that Entire Net-Work can listen on multiple ports simultaneously.

KEEPALIV Parameter



This optional parameter allows you to maintain connections when there is no other traffic with the remote links. Valid values are "Y" or "N."

- When this value is set to "Y", it causes internal TCP messages to be sent periodically to all remote links, thus maintaining the connections when there is no other traffic with the remote links. The amount of time between messages is determined by an initialization parameter in the TCP stack.
- When this value is set to "N", internal TCP messages are no longer sent periodically and the connections are not maintained.

The default for this parameter is "N".

KEEPALIV can also be set for individual remote links. For more information, read about the KEEPALIV parameter associated with the **TCPX LINK statement**.

MULTSESS Parameter

This optional parameter determines whether a connect request from a host that has an active connection is treated as a new link. This parameter can be modified when the line driver is open or closed.

A value of "Y" indicates that the connect request is treated as a new link; a value of "N" indicates that the connect request is rejected.

The default for this parameter is "Y".

NUMUSERS Parameter

```
NUMUSERS = { number | 100 }
```

This parameter specifies the estimated maximum number of concurrently active clients. For performance reasons, a table of individual client entries is preallocated based on this number. During the Entire Net-Work session, if the number of active clients is exceeded, the table is automatically expanded by 50% of the current value. The size of each entry in the table is 256 bytes. The minimum value is 10, maximum is 32767. The default is 100.

Note: This parameter can only be altered when the driver is closed.

OPTIONS1 Parameter

```
OPTIONS1 = (n,n,n,n,n,n,n,n,n,n)
```

This optional parameter allows up to ten numeric API-specific options to be set. The values can be modified when the line driver is open or closed. There are no default values.

OPTIONS2 Parameter

```
OPTIONS2 = (x,x,x,x,x)
```

This optional parameter allows up to five alphanumeric API-specific options to be set. The values can be modified when the line driver is open or closed. There are no default values.

PSTATS Parameter

```
PSTATS = { Y | N }
```

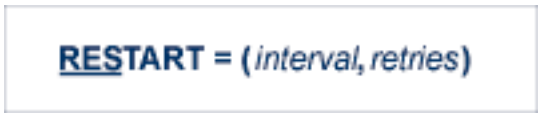
This optional parameter determines whether or not statistics are printed.

A value of "Y" indicates that statistics should be printed to DDPRINT when the statistics interval expires; a value of "N" indicates that the statistics should not be printed.

This parameter does not affect the STATS command and can be modified when the driver is open or closed.

The default for this parameter is "N".

RESTART Parameter



This optional parameter specifies the retry interval in seconds (*interval*) and the number of retries (*retries*) that Entire Net-Work will attempt to reopen the access method with the API after a shutdown due to a failure. The RESTART parameter can be modified when the line driver is open or closed.

If RESTART is not specified, or *interval* is specified as zero, no retry is attempted. By specifying (*retries*) as zero, an infinite number of retries can be requested.

The RESTART parameter is particularly useful with the Simple Connection Line Driver when Entire Net-Work is started at IPL and communication with the API is unsuccessful because TCP/IP is not yet fully initialized. Using this parameter, you can instruct Entire Net-Work to reopen the TCP/IP session, thereby giving TCP/IP sufficient time to become active.

The TIMER parameter on the NODE statement affects the RESTART parameter (see the section *Entire Net-Work NODE Statement* in the *Entire Net-Work Reference Guide*.) The retry interval should not be less than the TIMER parameter, and should be a multiple of this value. If a retry interval other than zero is specified that is less than the value of the TIMER parameter, the TIMER value is used instead.

RSTATS Parameter

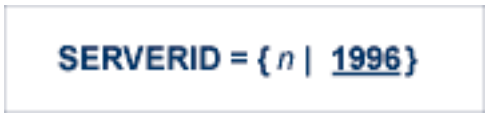
This optional parameter determines whether or not statistics are reset.

A value of "Y" indicates that statistics should be reset when the statistics interval expires; a value of "N" indicates that the statistics should not be reset.

The RSTATS parameter can be modified when the line driver is open or closed.

The default for this parameter in "N".

SERVERID Parameter



This optional parameter specifies a well known port number used by Entire Net-Work while awaiting connection requests from participating Entire Net-Work partners. Values may range from 1 to 65535. The SERVERID parameter can be modified only when the line driver is closed.

When specified in a DRIVER statement, the SERVERID parameter specifies the port number of the Entire Net-Work being initialized. If SERVERID is not specified for a link, the SERVERID specified for the driver is used as the default port for the link.

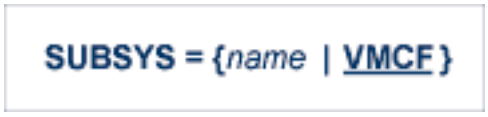
The default for this parameter is 1996. Only the DRIVER statement has a SERVERID parameter.

STATINT Parameter

This optional parameter specifies the amount of time, in seconds, before statistics are automatically printed or reset. The default is 3600. The STATINT parameter can be modified when the line driver is open or closed.

Acceptable values range from 1 to 2147483647. Any value outside this range is in error.

SUBSYS Parameter



This parameter specifies the name of the subsystem to be accessed by the API routines that use subsystem control blocks in interaddress space communications. The default value is VMCF. The SUBSYS parameter can be modified only when the line driver is closed.

In a z/OS environment, the IBM API routines communicate to the system address space by locating the subsystem control table and retrieving the information required to perform cross-memory communication. If the subsystem is specified incorrectly, the driver is not able to perform its open processing and no connections are possible.

TRACE Parameter

```
TRACE = { Y | N }
```

This parameter indicates whether tracing for this line driver should be active (Y) or not (N). When tracing is activated, trace information is placed in the trace table. The default is N (no). The TRACE parameter can be modified when the line driver is open or closed.

This is equivalent to specifying `TRACE=linedriver-code` or `TRON=linedriver-code` in the NODE statement (for example, `TRACE=CTCA`).

TRACELEV Parameter

```
TRACELEV = ( Y | N, Y | N, Y | N, Y | N, Y | N, Y | N, Y | N, Y | N, Y | N, Y | N )
```

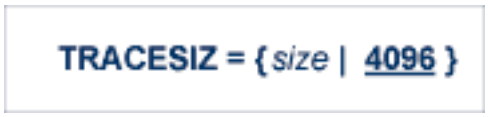
This optional parameter specifies the levels of tracing that the line driver will perform. It is a series of flags that determine which events are traced. The TRACELEV specification must be enclosed in parentheses. For example:

```
TRACELEV=(N,N,N,N,N,N,N,N,N,N)
```

Trace levels are positional within the parameter syntax and are set using Y (Yes) or N (No). It is recommended that all settings within the TRACELEV parameter be N. If your system experiences problems, contact your Software AG technical support representative for the settings that produce the appropriate trace information. The TRACELEV parameter can be modified when the line driver is open or closed.

Note: The tracing information provided is sent to the DDPRINT data set. In addition to setting the TRACELEV flags, the trace must also be turned on using either the DRIVER statement parameter `TRACE=Y` or the operator command `TRACE=linedriver-name`. Tracing dramatically affects the overall performance and throughput of Entire Net-Work.

TRACESIZ Parameter



This optional parameter specifies the size, in bytes, of the driver-specific trace table.

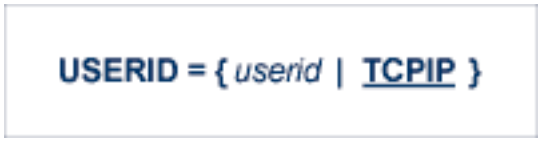
This parameter is also used as the default size of the link specific trace table when the LINK statement does not include a TRACESIZ specification.

Valid values can range from 4096 to 4194304. A value less than 4096 is reset to 4096; a value greater than 4194304 is reset to 4194304.

The TRACESIZ parameter can be modified only when the line driver is closed.

The default for this parameter is "4096".

USERID Parameter



This parameter's value can be modified only when the line driver is closed.

The USERID parameter specifies the name of the started task, job, or virtual machine in which the IBM TCP/IP protocol stack is running. The value is 1-8 characters. The default value is TCPIP.

14 TCPX LINK Statement

The LINK statement and its parameters are used to define the characteristics of the remote client. With the Simple Connection Line Driver, links are not normally predefined; they are dynamically allocated as clients initiate communication. However, links may be predefined to override defaults or provide some control over clients.

LINK Statement Format

The TCPX LINK statement has the following format:

```
LINK linkname TCPX  [ADJHOST=Internet-host-name, ]
                    [INETADDR=(n1.n2.n3.n4) , ]
                    [KEEPALIV={ Y | N } , ]
                    [MULTSESS={ N | Y } , ]
                    [PSTATS={ N | Y } , ]
                    [RSTATS={ N | Y } , ]
                    [SAF={ Y | L | N } , ]
                    [SENDTIME={ time | 90 } , ]
                    [STATINT={ interval | 3600 } , ]
                    [TRACESIZ=size ]
```

For more information about syntax conventions and rules used in this section, read ***Conventions***.

Modifying the LINK Statement Parameters

The LINK statement parameters are read from a sequential file during system startup, and can be modified after startup using the `ALTER` operator command. Some parameters can be modified when the link is open or closed. Others can be modified only when the link is closed. Read about the `ALTER` and `CLOSE` commands in the section ***TCPX Operator Commands***. The open/closed requirement for each parameter is included in the parameter descriptions.

LINK Statement Parameters

This section describes all of the parameters that can be used for the TCPX LINK statement.

linkname Parameter

```
linkname
```

The required *linkname* parameter specifies the name by which this link is to be known. It is positional, and must be specified immediately after the LINK keyword and immediately before the driver name (TCPX); the link name must be unique on the node. All operator commands affecting the link must specify this name.

If the link name begins with the characters "MODEL", the link is defined as a model link. See the section ***Model Links***.

TCPX Parameter

```
TCPX
```

TCPX is required and specifies the protocol name that defines the driver associated with this link. It must be the same as the value specified for the **DRVNAME parameter** in the **TCPX DRIVER statement**. In most cases, this value is "TCPX". If DRVNAME is changed to a value other than "TCPX", this parameter must also be changed.

ADJHOST Parameter

```
ADJHOST=Internet-host-name
```

This optional parameter specifies the Internet host name of a node with which a connection is to be established. Its value can be 1 - 255 characters. The ADJHOST parameter can be modified only when the link is closed.

The ADJHOST parameter uses Domain Name Services (DNS), as follows:

- The GetHostByName function is used to determine the IP address of a host name specified with ADJHOST. IP address is used both for connecting to another node and for locating the link for an incoming connection.
- The GetHostByAddr function is used to determine the host name of a node that is trying to connect to this node. This is necessary when the IP address of a host name specified with ADJHOST changes after the link has been opened.

Software AG recommends the use of the ADJHOST parameter for sites that assign IP addresses via the DHCP protocol. Entire Net-Work will use the GetHostByName function for every outgoing connection on nodes that have ADJHOST specified as long as INETADDR is not specified.

The following table lists the APIs that support Domain Name Services:

API	GetHostByName	GetHostByAddr
HPS	Yes	Yes
OES	Yes	Yes

For performance reasons, Software AG recommends that all LINK statements containing an ADJHOST value be defined after the LINK statements containing an INETADDR specification. If neither of these parameters is specified, the link is not usable. If both are specified, the INETADDR parameter takes precedence.

INETADDR Parameter

```
INETADDR=(n1.n2.n3.n4)
```

This optional parameter specifies the IP (Internet Protocol) address of the remote host associated with this link. The INETADDR parameter can be modified only when the link is closed.

IP address is used both for connecting to another node and for locating the link for an incoming connection. It is provided to Entire Net-Work in the form of INETADDR=(n1,n2,n3,n4) or INETADDR=(n1.n2.n3.n4) where each value represents 8 bits of the 32-bit IP address. Acceptable values are between 0 and 255 and may be separated by commas or periods. For example:

```
INETADDR=(157,182,17,20) ↩
```

or

```
INETADDR=(157.182.17.20) ↩
```

Each host on the INTERNET is assigned a unique IP address which is used by the IP and higher level protocols to route packets through the network. The IP address is logically made up of two parts: the network number and the local address. This IP address is 32 bits in length and can take on different formats or classes. The class defines the length (number of bits) of each part. There are four classes (A, B, C, and D); the class is identified by the allocation of the initial bit.

For performance reasons, Software AG recommends that all LINK statements containing an INETADDR value be defined before the LINK statements containing an ADJHOST specification. If neither of these parameters is specified, the link is not usable. If both are specified, the INETADDR parameter takes precedence.

KEEPALIV Parameter

```
KEEPALIV={Y | N}
```

KEEPALIV=Y (Yes) causes internal TCP messages to be sent periodically to the remote node, thus maintaining the connection when there is no other traffic with the node. The amount of time between messages is determined by an initialization parameter in the TCP stack. If no KEEPALIV value is specified for the link, it defaults to the KEEPALIV value on the **DRIVER statement**.

MULTSESS Parameter

```
MULTSESS={N | Y}
```

This optional parameter determines whether a connect request from a host that has an active connection will be treated as a new link (Y), or a reconnection of an existing link (N). The default value is the value specified for the MULTSESS parameter in the **TCPX DRIVER statement** (N or Y, with Y as the default). The MULTSESS parameter can be modified when the link is open or closed.

PSTATS Parameter

```
PSTATS={N | Y}
```

This optional parameter determines whether or not (Y or N) statistics are printed to DDPRINT when the statistics interval expires. The default value is the value specified for the PSTATS parameter in the **TCPX DRIVER statement** (for which the default is N). This parameter does not affect the `STATS` operator command. The PSTATS parameter can be modified when the link is open or closed.

RSTATS Parameter

```
RSTATS={N | Y}
```

This optional parameter determines whether or not (Y or N) statistics are automatically reset when the statistics interval expires. The default value is the value specified for the RSTATS parameter in the **TCPX DRIVER statement**. The RSTATS parameter can be modified when the link is open or closed.

SAF Parameter

```
SAF={Y | L | N}
```

If SAF=Y or SAF=L is specified, Entire Net-Work will call the SAF Interface for all incoming requests on this link; failure to load the Interface is considered a security violation and Entire Net-Work will shut down. If SAF=L, the calls are traced and the output directed to DDPRINT. An error code is transmitted to the user if access to SAF is denied. The SAF parameter can be modified when the link is open or closed. The default value is N (No).

SENDTIME Parameter

```
SENDTIME={time | 90}
```

This optional parameter specifies the time (in seconds) that the Simple Connection Line Driver allows for a send to complete. When this time is exceeded, the line driver writes a message to the operator console indicating a possible error condition on the remote node. The connection is considered severed and link disconnect processing is initiated. The default value is 90 seconds. The SENDTIME parameter can be modified when the link is open or closed.

STATINT Parameter

```
STATINT={ interval | 3600}
```

This optional parameter specifies the amount of time, in seconds, before statistics are automatically printed or reset. Acceptable values are 1 - 2147483647. Any value outside this range is in error. The default value is 3600. The STATINT parameter can be modified when the link is open or closed.

TRACESIZ Parameter

```
TRACESIZ=size
```

This optional parameter specifies the size of the TCP/IP link specific trace table. Value can be 4096 - 4194304. A value less than 4096 is reset to 4096. A value greater than 4194304 is reset to 4194304. The default value is the value specified for the TRACESIZ parameter in the **TCPX DRIVER statement**. The TRACESIZ parameter can be modified only when the link is closed.

15 Simple Connection Line Driver Operator Commands

Entire Net-Work's Simple Connection Line Driver has the ability to process operator commands that are directed to a specific link or directly to the driver.

Operator Command Syntax

Under z/OS, the Simple Connection Line Driver operator commands have the following format:

F NETWORK, TCPX *target cmd*

The following table describes this syntax.

Syntax Representation	Description
TCPX	Informs Entire Net-Work that the command is destined for the Simple Connection Line Driver. If more than one **TCPX DRIVER statement** exists, use the name specified on the DRVNAME parameter of the **DRIVER statement** instead of TCPX.
target	A value that informs Entire Net-Work what the target of the command is, as follows: ■ Specify an asterisk (*) if the target is all links. ■ Specify the DRVCHAR value ("#" is the default) if the target is the driver itself (see the DRVCHAR parameter on the ***TCPX DRIVER Statement***). ■ Specify the link name if the target is a specific link.
cmd1, *cmd2*, and *cmdx*	The operator commands to be carried out. Multiple commands can be specified in a single command statement. When the `ALTER` command is specified, it must be the last command in the statement, because everything following the `ALTER` command is treated as a DRIVER or LINK statement parameter. One or more DRIVER or LINK statement parameters must be specified.

Examples

The following are examples of Simple Connection Line Driver operator commands:

```
F NETWORK,TCPX * CLOSE
```

```
TCPX # STATS
```

Driver Commands

The Entire Net-Work Simple Connection Line Driver supports the commands listed in the following table when the target is the driver. The underlined portion of the command is the minimum abbreviation.

Command	Action
ALTER *driver-parms*	Dynamically changes the driver configuration. The ALTER command is followed by the driver configuration parameters to be altered. The driver configuration parameters are the same as those specified in the DRIVER statement. For example: `TCPX # ALTER ACCEPTUI=Y`
CLOSE	Disconnects and closes all links that are connected to other nodes. Releases all resources held by the driver as well as all open links. Closes the driver.
OPEN	Reopens the driver after it is closed with the CLOSE operator command or because of an access method failure. Allocates all the resources needed by the driver to communicate with TCP/IP. Also attempts to resolve any unresolved host names.
RESET	Resets all statistics for the driver. Statistics are printed only if the STATS command precedes the RESET command.
SHOW	Displays the current configuration of the driver. The current configuration is always shown automatically following an ALTER command.
SNAP	Causes all control blocks specific to the link to be snapped (printed in hexadecimal). Driver-specific control blocks and Entire Net-Work specific control blocks are not snapped.
STATS	Causes the immediate printing of statistics and restarts the statistics interval. This command has no effect on the next automatic printing of statistics. To print and reset statistics, specify RESET immediately after the STATS command. For example: `TCPX # STATS RESET`
STATUS	Displays the current status of the driver as well as a count of messages received and sent.
TRACE	Causes the Simple Connection Line Driver to format and print the driver-specific trace table. The trace table is formatted and printed in hexadecimal automatically when the SNAP command is processed.
USERS	Displays the Adabas user ID in character and hexadecimal formats, the Context ID and Context Verifier values (these are part of the internal message header and can be used to help identify the client), and the number of database calls received for the client.

Note: When the driver is closed, it does not recognize the commands CLOSE, STATS, or RESET.

Link Commands

The Entire Net-Work Simple Connection Line Driver supports the commands listed in the following table when the target is a link or all links. The underlined portion of the command is the minimum abbreviation.

Command	Action
ALTER *link-parms*	Dynamically changes the link configuration. The ALTER command is followed by the link configuration parameters to be altered. The link configuration parameters are the same as those specified on the LINK statement. For example: `TCPX LINK1 ALTER ADJHOST=DALLAS`
CLOSE	Disconnects the link if it is connected to another node and releases all resources held by the link.
DISCONNECT	Starts the disconnect sequence for the target link(s). If the link is already disconnected or is in the process of disconnecting, the command is ignored.
LOGLON *linkname*	Turns on selective logging for the specified link.
LOGLOFF *linkname*	Turns off selective logging for the specified link.
OPEN	Allocates all the resources needed by the link to communicate with TCPX. Does not initiate a connect to the remote node. The status of the link displayed via the SHOW operator command is not affected by the OPEN request.
RESET	Resets all statistics for the link. Statistics are printed only if the STATS command precedes the RESET command.
RESUME	Restarts processing on a link that was temporarily stopped due to a SUSPEND command.
SHOW	Displays the current configuration of the link. The current configuration is always shown automatically following an ALTER command.
SNAP	Causes all link-specific control blocks and the link-specific trace table to be snapped (printed in hexadecimal). Driver-specific control blocks and Entire Net-Work-specific control blocks are not snapped.
STATS	Causes the immediate printing of statistics and restarts the statistics interval. This command has no effect on the next automatic printing of statistics. To print and reset statistics, specify RESET immediately after the STATS command. For example: `TCPX LINK1 STATS RESET`
STATUS	Displays the current status of the link as well as a count of messages received and sent.
SUSPEND	Temporarily stops all processing on a link. Processing can be restarted with the RESUME command.

Command	Action
TRACE	Causes the link-specific trace table to be formatted and printed. The trace table is formatted and printed in hexadecimal automatically when the SNAP command is processed.
USERS	Displays the Adabas user ID in character and hexadecimal formats, the IP address for the link, the Context ID and Context Verifier values (these are part of the internal message header and can be used to help identify the client), and the number of database calls received for the client.

16 Model Links

The Simple Connection Line Driver supports dynamically added links, thus reducing the time required to set up and maintain the TCPX LINK statements. Model links can be coded and used to define new links as they are added.

The Simple Connection Line Driver is permitted to add links dynamically if ACCEPTUI=Y is coded on the DRIVER statement. A new link block is created and is used to control all further communications on the link. The link block can be initialized with default values that will be applied to each new link. Alternatively, one or more model links can be defined to override the values contained in the link block.

The model link statement is identical to other LINK statements, except that the link name begins with the characters 'MODEL'. Most of the model link parameters, such as PSTATS and RSTATS, are copied into the dynamically built link block. Some parameters, such as INETADDR, are not copied because they are truly link-specific.

17 Simple Connection Line Driver Statistics

The Entire Net-Work Simple Connection Line Driver issues API calls to communicate with TCP. To help tune each link and the driver itself, the Simple Connection Line Driver provides the statistics shown below. The statistics for a link and the driver are identical, with the exception of the title line (a).

```
     +-------------------------------------------------------------------------+
(a)  + Statistics For Driver   TCPX  Period     0:02:42 (    162.325 Secs)  +
     + -------------  ---Bytes----  --Messages--  -Api Calls--  ----------- +
(b)  + Writes             496.394K         3,412         3,412  Total       +
(c)  +                      3.063K            21            21  Per Second  +
(d)  + Reads                0.000K             0         3,451  Total       +
(e)  +                      0.000K             0            21  Per Second  +
     + -------------  ---Total----  ----Task----  ---Other----  ----------- +
(f)  + Write Cmd's                0             0             0  Total       +
(g)  +                            0             0             0  Per Second  +
(h)  + Read Cmd's                 0             0             0  Total       +
(i)  +                            0             0             0  Per Second  +
     +-------------------------------------------------------------------------+
```

This multiple line display is produced when the STATS operator command is issued either for the TCPX driver or its links. This display is also produced when the automatic statistics interval expires and the PSTATS=Y is specified in the TCPX DRIVER or LINK statement. Values are displayed and updated asynchronously; therefore, the totals displayed may not always be accurate. The contents are as follows:

Line	Shows the . . .
a	name of the link or driver and the length of time since statistics were last reset or the link was last connected. Length of time is displayed in *hours:minutes:seconds* and in *seconds:milliseconds*.
b	cumulative number of bytes and messages written, and the cumulative number of API calls.
c	average number of bytes and messages written, and the average number of read API calls per second.
d	cumulative number of bytes and messages read, and the cumulative number of read API calls.

Line	Shows the . . .
e	average number of bytes and messages read, and the average number of read API calls per second.
f	cumulative number of WRITE commands that occurred. The total number of WRITEs is equal to the number of WRITEs from the Entire Net-Work task plus the average number of WRITEs from asynchronous routines.
g	average number of WRITE commands that occurred per second. The total average number of WRITEs is equal to the average number of WRITEs from the Entire Net-Work task plus the average number of WRITEs from asynchronous routines.
h	cumulative number of READ commands that occurred. The total number of READs is equal to the number of READs from the Entire Net-Work task plus the number of READs from asynchronous routines.
i	average number of READ commands that occurred per second. The total average number of READs is equal to the average number of READs from the Entire Net-Work task plus the average number of READs from asynchronous routines.

18 Connecting to UES-Enabled Adabas Databases

Prior to Adabas version 7, Entire Net-Work converted all data for mainframe Adabas when necessary from ASCII to EBCDIC. Starting with version 7, Adabas is delivered with its own data conversion capability (module LNKUES); that is, Universal Encoding Support (UES). Entire Net-Work detects when it is connected to a target database with UES support and passes the data through to Adabas without converting it.

In order for UES support to work, various ADALNK modules must be linked to the Adabas UES module, LNKUES. LNKUES converts data in the Adabas buffers and byte swaps, if necessary, depending on the data architecture of the caller.

Note: If Adabas versions prior to 7.4 are used by Entire Net-Work, you must create a UES-enabled ADALNK module, as described in this chapter.

Overview of UES Support

This section provides a general overview of the UES support provided in Adabas. For detailed information about UES support in Adabas, refer to the Adabas installation documentation for the operating system you are using.

In order for UES support to work, various ADALNK modules and tables must be linked to the Adabas UES module, LNKUES. LNKUES converts data in the Adabas buffers and byte-swaps, if necessary, depending on the data architecture of the caller.

This section covers the following topics:

- UES-Linked Load Modules
- UES Translation Tables
- Job Steps
- Calling LNKUES
- Connection Possibilities

UES-Linked Load Modules

The ADALNK load modules that have been linked with module LNKUES vary, depending on the operating system and environment you are running. For a complete list of the modules that have been linked, read the UES sections of the appropriate Adabas installation documentation.

Note: If Adabas versions prior to 7.4 are used by Entire Net-Work, you must create a UES-enabled ADALNK module, as described in ***Verify Required ADALNK Module Available***, elsewhere in this section.

UES Translation Tables

Two standard translation tables are provided with Adabas UES support:

- ASC2EBC: ASCII to EBCDIC translation
- EBC2ASC: EBCDIC to ASCII translation

The Adabas and Entire Net-Work translation table pairs are provided in the appropriate Adabas installation documentation.

You can use the supplied translation tables, or you may prepare your own customized translation tables, reassemble them , and link them with the LNKUES module. Using your own customized translation tables should only be necessary if you require the use of some country-specific character other than the standard A-Z, a-z, or 0-9 characters in the additions 1 (user ID) or additions 3 field of the control block. For detailed information on using the default or customized translation tables, refer to the UES sections of the appropriate Adabas installation documentation.

If you prefer to use the same translation tables that are used in Entire Net-work, change the COPY statements in ASC2EBC and EBC2ASC from UES2ASC and UES2EBC to NW2ASC and NW2EBC, respectively. After modifying the translation tables, be sure to (re)assemble them and link them with the delivered LNKUES module. The sample jobs referenced in *Connecting to UES-Enabled Databases through Entire Net-Work* include steps that reassemble and link the translation tables with LNKUES.

Job Steps

Job library members are provided with Adabas for each operating system it supports to assemble and link the appropriate modules with the UES components. For more information, read the UES sections of the appropriate Adabas installation documentation.

Calling LNKUES

On all platforms, LNKUES receives control before UEXITB for UES requests and after UEXITA for UES replies.

Connection Possibilities

UES-enabled databases are connected to machines with different architectures through Entire Net-Work. These connections methods are described elsewhere in this section.

Environment Requirements

To support UES-enabled databases, be sure that your environment meets the requirements described in this section.

Database Requirements

The Adabas database must be UES-enabled. For complete information read about database maintenance tasks in the Adabas DBA tasks documentation and about the ADACMP and the ADADEF utilities in the Adabas utilities documentation.

In general, to UES-enable an Adabas database:

- Specify `MODIFY UES=YES` in the Adabas ADADEF utility settings for each target database.

Data Set Requirements

Make sure that the internal product libraries described in this section are loaded and concatenated correctly.

Required Internal Product Libraries

Software AG internal product libraries that are required if you intend to enable a database for universal encoding service (UES) support are now delivered separately from the product libraries. For UES support, the following libraries must be loaded and included in the STEPLIB or LIBDEF concatenation:

```
APS272.MVSLDnn
```

where *nn* is the load library level. If the library with a higher level number is not a full replacement for the lower level load library(s), the library with the higher level must precede those with lower numbers in the steplib concatenation.

Also for UES support, the following library must be loaded and included in the session execution JCL:

```
ADABAS.Vvrs.ADAvrs.MVSECOn ↩
```

This library includes all supported code pages. For more information about the supported code pages, read about *Supplied UES Encodings* in your Adabas documentation.

Adabas JCL Updates

If you intend to enable your database for universal encoding service (UES), the startup job for the Adabas nucleus must be updated as described in this section.

- The MVSLD*nn* internal product libraries must be concatenated in the STEPLIB or LIBDEF. The following is an example of such a STEPLIB concatenation:

  ```
  //STEPLIB DD DISP=SHR,DSN=ADABAS.Vvrs.ADAvrs.MVSLOAD
  //        DD DISP=SHR,DSN=ADABAS.Vvrs.APSvrs.MVSLDnn
  ```

  where *nn* is the load library level. If the library with a higher level number is not a full replacement for the lower level load library(s), the library with the higher level must precede those with lower numbers in the steplib concatenation.
- Also for UES support, the following ECS objects data set must be loaded and included in the session execution JCL:

  ```
  //DDECSOJ DD DISP=SHR,DSN=ADABAS.Vvrs.ADAvrs.MVSECOn
  ```

  Note: The data set DDECSMF (messages) previously required for UES support no longer exists and reference to it needs to be deleted from your JCL. Likewise, the CONFIG DD "dummy" data set is no longer needed.

Disk Space Requirements for Internal Product Data Sets

The minimum disk space requirements on a 3390 disk for the internal product libraries delivered with Adabas is as follows:

Libraries	3390 Cylinders	3390 Tracks	Directory Blocks
APS272.MVSLD00	5	75	55

SYSPARM Requirements

To support UES, you need to add SYSPARM statements and parameters to your session execution JCL, as follows:

```
//SYSPARM DD*
SYSTEM_ID=ADAAPS
ABEND_RECOVERY=NO
THREAD_ABEND_RECOVERY=NO
```

These SYSPARM statements and parameters are required for the APS internal product.

Verify Required ADALNK Module Available

Ensure that the ADALNK module in Entire Net-Work's STEPLIB or LIBDEF is UES-enabled. If Adabas or Adabas Limited Load Library (WAL) Version 7.4 or later are used by Entire Net-Work, then ADALNK is UES-enabled by default.

If Adabas or WAL version 8 or later are used by Entire Net-Work, you must create a UES-enabled ADALNK module by completing the steps in *Connecting to UES-Enabled Databases through Entire Net-Work*.

Connecting to UES-Enabled Databases through Entire Net-Work

If you are using Adabas 7.4 or later, there are no additional steps to perform to connect to UES-enabled databases through Entire Net-Work; the ADALNK module has been UES-enabled for you.

If you are using a version of Adabas prior to 7.4, you must create a UES-enabled ADALNK module to use when connecting to UES-enabled databases through Entire Net-Work.

Regardless of the Adabas version you are using, if you have altered the translation tables, you will need to perform the step described in this section (depending on the platform) that assembles and links the updated translation tables into ADALNK.

In all cases, whenever you alter ADALNK, you must be sure to make the updated module available to Entire Net-Work.

UES-Enabling ADALNK on z/OS Systems

To create a UES-enabled ADALNK module on z/OS systems:

1 Modify the standard batch ADALNK:

```
&UES SETB 1
```

2 Assemble and link the modified batch ADALNK with the translation tables and LNKUES. You can use the sample JCL found in MVSJOBS member ALNKLNK7 in the WAL libraries. Make sure you:

- Provide all necessary job card information.
- Check the symbolic parameter value for version, revision level, and SM level (*vrs*). It must reflect the level of your Adabas source and load libraries.
- Check the data set names for SYSLIB, SYSIN, SYSLMOD, and SYSLIN in the SAGASM and LINKALL inline procedures.

3 Once it is successfully linked, make ADALNK available to Entire Net-Work's job STEPLIB concatenation list.

For complete instructions on UES-enablement in Adabas, refer to the UES sections of your Adabas documentation.

19 Estimating Entire Net-Work Storage Requirements

Given the complexity of today's data processing environments, it is almost impossible to provide methods to predict the exact storage requirements of a software product.

The following tables provides rough estimates about the fixed storage requirements of Entire Net-Work and its various components, ignoring operating system-related storage requirements, which typically vary from installation to installation.

Table 1 contains the amounts of storage obtained from the operating system based on parameter specification or appropriate defaults. It does not include storage areas that are directly related to the operating system, such as operating system control blocks, I/O-related buffers, and control blocks (except where they are part of Entire Net-Work program modules or data areas).

Table 2 contains the amounts of storage obtained from the Entire Net-Work buffer pools by the control module and the various line drivers.

Table 3 contains special storage requirements of the line drivers (such as special common system storage areas) in the various operating system environments.

Table 1: Storage Areas Obtained from System

Storage Area		Platform			
		z/OS	VSE	VM	BS2000/OSD
Request queue: (NC parameter+1)*192		AS(X)	SYS/Part	Virt.M	Comm.Pool
Attached buffers: (NAB parameter*4112)		AS(X)	SYS/Part	Virt.M	Comm.Pool
Entire Net-Work buffer pools*	Asynchronous buffers	AS(X)	Part	Virt.M	AS(X)
	Long-term buffers	AS	Part	Virt.M	AS
	Short-term buffers	AS(X)	Part	Virt.M	AS(X)
	Page-fixed buffers	AS	Part	Virt.M	AS
Entire Net-Work trace table		AS(X)	Part	Virt.M	AS(X)
Entire Net-Work control blocks	general	AS	Part	Virt.M	AS
	Node	48	48	48	48
	Target	32	32	32	32
	Path	32	32	32	32
	CTCA DRIVER	544	---	---	---
	DCAM DRIVER	---	---	---	848
	IUCV DRIVER	---	4KB	---	---
	TCPI DRIVER	4KB	4KB	4KB	4KB
	TCPX DRIVER	4KB	4KB	4KB	4KB

Storage Area		Platform			
		z/OS	VSE	VM	BS2000/OSD
	VTAM DRIVER	4KB	4KB	4KB	4KB
	XCF DRIVER	2048	---	---	---
	CTCA LINK	992	---	---	---
	DCAM LINK	---	---	---	56
	IUCV LINK	---	168	168	168
	TCPI LINK	1KB	1KB	1KB	1KB
	TCPX LINK	1KB	1KB	1KB	1KB
	VTAM LINK	256	256	256	256
	XCF LINK	2048	---	---	---
ADAIOR data areas	general	AS	Part	Virt.M	AS
	(for trace table, ECB list, etc.)	about 2KB	about 2KB	about 2KB	about 2KB

Abbreviation	Meaning
AS	from address space (private, below 16MB if XA or XS)
AS(X)	from address space (private, above 16MB if XA or XS)

Table 2: Storage Obtained from Entire Net-Work Buffer Pools

Statistic	Buffer Pool Types			
	Asynch	Long-term	Short-term	Page-fixed
Segment size	64	64	512	2KB or 4KB
Control module buffer pool usage		UB	MSG RPLY	
Queue manager buffer pool usage			BLK	BLK

Abbreviation	Meaning
BLK	Storage for outgoing transmission blocks (after compression and blocking), from short-term pool or page-fixed pool, depending on line driver requirements. Storage requirements for one transmission block include, in addition to the messages contained, 48 bytes for a transmission block header.
MSG	All messages sent or received; output messages kept until acknowledged by the access method, input messages kept until processed.

Abbreviation	Meaning
	The size of a message can be computed in the following way: 56 bytes for a message header + maxpath * 2 bytes for a node stack + 128 bytes for UB, ACB, etc. + size of FB, RB, SB, VB, IB to send or receive
RPLY	A reply buffer for each user request for a target on this node if the information returned by the target will not fit into the original message buffer (that is, if a large record buffer or ISN buffer is to be returned to the user).
UB	(only if 31-bit mode:) 64 bytes per user request for a target on this node, for the duration of the Adabas call.

Table 3: Special Storage Requirements of Line Drivers

Driver	Special Storage Requirements
TCPX	NUMUSERS*256 is initially allocated from buffer pool storage for the Active Client Table (ACT). This value may dynamically expand if required.

Note: In addition to the storage estimates shown in the table, approximately 250KB storage is required for executable code.

Index

N

O

P

R

S

T

U

V