

# **Entire Net-Work**

## **Concepts and Facilities**

Version 6.6.1

October 2023

This document applies to Entire Net-Work Version 6.6.1 and all subsequent releases.

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

Copyright © 1994-2023 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

**Document ID: WCPMF-CONCEPTS-661-20231014**

## Table of Contents

Entire Net-Work Concepts .....	v
1 About this Documentation .....	1
Document Conventions .....	2
Online Information and Support .....	2
Data Protection .....	3
2 Introduction to Entire Net-Work .....	5
How Entire Net-Work Operates .....	6
Entire Net-Work Components .....	8
Summary of Entire Net-Work Features .....	9
3 Designing Your Entire Net-Work Configuration .....	13
Network Design .....	14
Entire Net-Work Components .....	14
Topologies .....	14
Expanding the Network Configuration .....	16
Redundancy .....	17
Weighting .....	18
Traffic Considerations .....	19
Broadcasting .....	19
Client-Only Nodes .....	21
Number of Hops .....	21
4 Licensing Entire Net-Work .....	23
5 Starting Entire Net-Work .....	25
6 Entire Net-Work Security (z/OS only) .....	27
SAF-based Security Packages .....	28
NETSAF .....	28
Adabas SAF Security .....	28
7 Entire Net-Work Light .....	31
Overview .....	32
Use Cases .....	32
Installation .....	33
TCPL Driver Statement .....	36
TCPL Driver Operator Commands .....	45
Entire Net-Work Light Operator Commands .....	48
Messages Specific to Entire Net-Work Light .....	49



---

# Entire Net-Work Concepts

---

This document introduces you to Entire Net-Work and provides information about designing your Entire Net-Work configuration.

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

<i>Introduction to Entire Net-Work</i>	Explains how Entire Net-Work operates and describes product components and features.
<i>Designing Your Entire Net-Work Configuration</i>	Describes Entire Net-Work design considerations.
<i>Licensing Entire Net-Work</i>	Describes the Entire Net-Work licensing concept.
<i>Starting Entire Net-Work</i>	Describes how Entire Net-Work is started.
<i>Entire Net-Work Security (z/OS only)</i>	Describes the (z/OS only) security features available with Entire Net-Work
<i>Entire Net-Work Light</i>	Explains how Entire Net-Work Light operates.

---

# 1

## About this Documentation

---

■ Document Conventions .....	2
■ Online Information and Support .....	2
■ Data Protection .....	3

## Document Conventions

---

Convention	Description
<b>Bold</b>	Identifies elements on a screen.
Monospace font	Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.
<i>Italic</i>	Identifies:  Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
Monospace font	Identifies:  Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the   symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

## Online Information and Support

---

### Product Documentation

You can find the product documentation on our documentation website at <https://documentation.softwareag.com>.

In addition, you can also access the cloud product documentation via <https://www.software-ag.cloud>. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

### Product Training

You can find helpful product training material on our Learning Portal at <https://knowledge.softwareag.com>.



## Tech Community

You can collaborate with Software AG experts on our Tech Community website at <https://tech-community.softwareag.com>. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at <https://github.com/softwareag> and <https://hub.docker.com/publishers/softwareag> and discover additional Software AG resources.

## Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at <https://empower.softwareag.com>. Many services on this portal require that you have an account. If you do not yet have one, you can request it at <https://empower.softwareag.com/register>. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

## Data Protection

---

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.



## 2 Introduction to Entire Net-Work

---

■ How Entire Net-Work Operates .....	6
■ Entire Net-Work Components .....	8
■ Summary of Entire Net-Work Features .....	9

Entire Net-Work for BS2000, z/OS, and z/VSE provides transparent connectivity between client and server programs running on different physical or virtual machines, with potentially different operating systems and hardware architectures. The currently supported set of server programs includes Adabas, Entire System Server, EntireX Communicator, and any other software program that participates in cross-address communications defined by Software AG. A range of client programs are supported, including those written in Natural, the commonly used 4GL provided by Software AG, web-based applications such as Software AG's Jadabas and Tamino, and currently existing Adabas applications.

At its lowest level, Entire Net-Work accepts messages destined for targets or servers on remote systems and delivers them to the appropriate destination. Replies to these requests are then returned to the originating client application, without any change to the application. Entire Net-Work establishes these connections either through its line drivers or, on z/OS systems if the database is UES-enabled, through Entire Net-Work.

The method of operation and the location and operating characteristics of the servers are fully transparent to the user and the client applications. The servers and applications can be located on any node within the system where Entire Net-Work is installed and communicating. The user's view of the network targets and servers is the same as if they were located on the user's local node. Note that due to possible teleprocessing delays, timing of some transactions may vary.

## How Entire Net-Work Operates

---

Entire Net-Work provides transparent support for remote and distributed server processing by supporting the existing Adabas database interface. A user call to Adabas invokes the environment-specific Adabas Link Routine (ADALNK). This routine issues an interregion call to Adabas through the Adabas router (in z/OS and z/VSE, the router is the Adabas SVC). The router, in turn, locates the Adabas nucleus operating in a separate address space or partition, and adds the user call to the Command Queue (CQ). The Adabas nucleus then selects commands from the Command Queue and performs its normal processing.

Because there is no inter-system database communication, this configuration cannot support a user on one system and a database located on one or more other systems. The router was not designed to pass a request across the network to a remote database nucleus or other service on another system.

This section explains how Entire Net-Work solves this problem by simply extending the existing client/server interface:

- [Entire Net-Work Establishes the Connections](#)
- [Nodes Exchange Information](#)
- [The User Issues an Adabas Call](#)
- [Entire Net-Work Handles the Call](#)
- [Receiving Communicator Accepts the Message](#)

- Reply is Passed to the Requestor

## Entire Net-Work Establishes the Connections

Entire Net-Work establishes its connections using two possible mechanisms:

- An Entire Net-Work line driver can be installed on each machine. Line drivers establish a connection with each other through the appropriate access method services. This configuration supports two-way transfer of requests and replies between these two systems, as well as topology and server broadcasting. The line drivers available for Entire Net-Work are: CTCA, FCTC, IUCV, SSL, TCP/IP, TCPX, VTAM, and XCF.



**Note:** For more information about Encryption for Entire Net-Work (including the SSL line driver), contact your Software AG sales support representative. The documentation for Encryption for Entire Net-Work is delivered separately from the other Entire Net-Work documentation.

- Adabas UES-enabled databases can be connected through Entire Net-Work. For more information, read *Connecting to UES-Enabled Databases through Entire Net-Work*, in the *Entire Net-Work Administration Guide*.
- Adabas UES-enabled databases can be connected through ADATCP -- a direct TCP/IP link to the Adabas nucleus from web-based applications such as Software AG's Jadabas. For more information, read *Connecting to UES-Enabled Adabas Databases* and *Activating ADATCP*.

## Nodes Exchange Information

A node is defined by the Entire Net-Work NODE statement. There is one node per router and at least one router per machine. Each node in a network contains one copy of Entire Net-Work, which monitors any active servers (or targets) on that router and exchanges information with other Entire Net-Work nodes located on the same or on other machines. The Entire Net-Work nodes exchange information about targets accessible to them by means of broadcast messages. This information is maintained in tables by each Entire Net-Work node. Entire Net-Work also identifies itself to the system so that it receives all client requests for locally unavailable servers. Thus it can transport requests to remove servers or determine that a given server is currently not active anywhere. Client programs use server identifiers to direct their requests and are totally insensitive to the actual location of the servers.

## **The User Issues an Adabas Call**

To begin the process, the user issues an Adabas (or any other supported) call. The call is presented to the local router by the Adabas Link Routine, which finds that the requested service is not available locally. Instead of returning Response Code 148 to indicate "service not available", the router passes the call to the locally executing Entire Net-Work communicator.

## **Entire Net-Work Handles the Call**

After accepting the call, Entire Net-Work determines whether the requested service is available on the network. If it is not available, the local Entire Net-Work communicator returns Response Code 148 to the user to indicate "service not available".

If the target is active somewhere on the network, the call (including all required buffers) is packaged as a message and sent across the network to the target node by the line drivers. If necessary and the network topology allows, the message may be routed through many intermediate systems before it reaches its final destination.

## **Receiving Communicator Accepts the Message**

On the target node, the receiving communicator accepts the message and presents the contained call to the locally running service through the router. The actual server regards the Entire Net-Work call as equal to any other call issued from within its own local environment and returns any required reply in the normal manner through the local router.

## **Reply is Passed to the Requestor**

The router passes the reply information to the requestor, which is the local Entire Net-Work communicator, in the same way it passes reply information to a local request. The reply information is then packaged for the return trip to the originating node's communicator, which uses its local router to pass the reply to the user's request back to the client application.

## **Entire Net-Work Components**

---

Entire Net-Work is installed on each participating host or workstation system requiring client/server capability. The configuration for a given system includes the following components:

- an Entire Net-Work control module;
- control module service routines;
- any required line driver; and
- ADATCP -- a direct TCP/IP link to Adabas UES-enabled databases from web-based applications such as Software AG's Jadabas.

Each system with Entire Net-Work installed and running becomes a *node* in the network. Each node has one or more line drivers that define the node's identity on the machine or host where data can be received. Each line driver has one or more links that contain the driver identifier for sending data to other Entire Net-Work nodes on the same or other machines.

Entire Net-Work Each Entire Net-Work node maintains a *request queue* for incoming requests. This queue is similar to the command queue used by Adabas; it allows the node to receive Adabas calls from locally executing user/client programs, which Entire Net-Work then dequeues and transports to the nodes where the requested services reside.

Each local Entire Net-Work node also keeps track of all active *network services*, and therefore can determine whether the user's request can be satisfied or must be rejected. If the request can be serviced, the message is transmitted; otherwise, Entire Net-Work advises the calling user immediately with Response Code 148, just as the Adabas router would do for a local database request.

Actual network data traffic is controlled by Entire Net-Work *line drivers*, which are interfaces to the supported communications access methods, such as VTAM, IUCV, XCF, SSL, TCP/IP, and TCPX, or directly to hardware devices, such as channel-to-channel adapters (CTCAs or FCTCs). Each Entire Net-Work node contains only those line drivers required by the access methods active at that node. In addition, each line driver supports multiple connections to other nodes; this modular line driver design permits easy addition of new access method support to the system.

## Summary of Entire Net-Work Features

---

The following is an overview of Entire Net-Work features:

- **Distributed transaction processing**

With Software AG's Adabas Transaction Manager, a server for coordinating "two-phase commit processing" in distributed Adabas environments, users query and modify resources under the control of one or more database management systems (DBMSs) operating in one or more system images distributed physically across multiple Entire Net-Work nodes.

- **Operating system-independent architecture**

Like Adabas, Entire Net-Work consists of many operating system-independent routines, allowing faster and easier adaptation to new operating system versions.

- **Adabas compatibility**

Entire Net-Work uses Adabas-dependent service routines for the operating system interface as well as for interregion communication, thus avoiding incompatibility.

- **Adabas-like "look and feel"**

The similarity between Entire Net-Work and Adabas means that the job control statements for running Entire Net-Work are much like those needed to run Adabas. For example, the EXEC statement invokes the ADARUN program for Entire Net-Work just as it does for Adabas, and the ADARUN parameters for Entire Net-Work are a subset of Adabas parameters.

■ **Multiple communication access methods supported**

Access method support is implemented in the form of line drivers. If multiple access methods are needed on a node, supporting line drivers can be added without major reconfiguration of the node itself. Adding a new access method generally requires adding only the line driver module to the library and including the required configuration statements. With this implementation, a client request could be received from a VTAM partner and sent across to another machine using a channel-to-channel or TCP/IP connection for server processing.

■ **Access for UES-enabled mainframe databases**

Adabas UES-enabled databases can be connected through Entire Net-Work. For more information, read *Connecting to UES-Enabled Databases through Entire Net-Work*, in the *Entire Net-Work Administration Guide*.

■ **Automatic target status updating**

All targets and services establishing or terminating communication with the network cause status information to be broadcast to all nodes, eliminating the need to maintain or refer to database or target parameter files at a central location.

■ **Unique target ID enforcement**

Entire Net-Work enforces the Adabas requirement that each enterprise-wide target be assigned a unique target ID. (With Adabas, local targets that are introduced may have non-unique IDs.)

■ **Remote processing of client/server request**

A request can be made from within a Software AG or third party application client program to a server (typically Adabas) located on a remote system, as if the server were running locally with no client changes.

■ **One Entire Net-Work per node**

Allowing only one Entire Net-Work task on each node enforces control over the network topology by maintaining all required information in one place. This avoids confusion in network operation and maintenance. If, however, more than one Entire Net-Work task is required, this can be accomplished by installing additional routers.

■ **Single request queue for all remote targets**

Each Entire Net-Work node maintains only one request queue and one attached buffer pool for economical use of buffer storage.

■ **Transmission of relevant buffers only**

Entire Net-Work eliminates from transmission all buffers that are not required for the particular command. In addition, only those portions of the Record Buffer and ISN Buffer that have actually been filled by the server are returned to the user on a database reply. Software AG still recommends that you set the correct buffer sizes in the control block when coding direct calls; otherwise, unnecessary data may be transmitted.

■ **Selectable message blocking and compression**

Some Entire Net-Work line drivers allow specification of message blocking or message compression. In access methods where a large physical block size is possible, blocking can improve performance during peak message load times by reducing the number of transmissions and requests made to the access method service routines. Similarly, compressing messages reduces



transmission line loading and improves blocking statistics, but increases CPU consumption within Entire Net-Work.

■ **All buffer sizes allowed**

Buffer size support in Entire Net-Work is comparable to that in Adabas, ensuring that all buffer sizes that are valid for Adabas can also be transmitted to remote nodes.

■ **Entire Net-Work communication in heterogeneous systems**

Some line drivers support communication between systems with different hardware architectures, e.g., VTAM and TCP/IP. This allows for client/server communications to and from Entire Net-Work on Windows, OpenVMS and UNIX operating systems.

■ **User exit interface**

Some Entire Net-Work line drivers support an interface to an optional user exit. The user exit can encrypt or compress data before transmission and decrypt or decompress data after reception. It can decide to accept or reject a connection from a host that is not predefined to Entire Net-Work; when accepting such a connection, it has the ability to select the correct model link parameters.

■ **Model Links**

The "model" link facility allows users to code one or more model links with parameter values that serve as default values for many partners, instead of coding one LINK statement for each partner. As each partner connects, new control blocks are allocated and initialized from the model link.

■ **Additional operator commands**

Entire Net-Work's CTCA, FCTC, IUCV, VTAM, TCPI, TCPX , SSL, and XCF line drivers have the ability to process operator commands that are directed to a specific link or directly to the driver. Some driver and link parameters can be modified with the ALTER operator command while the driver or link is open, thus allowing dynamic reconfiguration of the network. Refer to the specific parameter description for information on possible restrictions about modifying the parameter using the ALTER command.



# 3

## Designing Your Entire Net-Work Configuration

---

■ Network Design .....	14
■ Entire Net-Work Components .....	14
■ Topologies .....	14
■ Expanding the Network Configuration .....	16
■ Redundancy .....	17
■ Weighting .....	18
■ Traffic Considerations .....	19
■ Broadcasting .....	19
■ Client-Only Nodes .....	21
■ Number of Hops .....	21

This section describes network design considerations.

## Network Design

---

Network design is critical to providing the best response time to client applications. Client/server applications tend to multiply rapidly, causing networks to expand without following a carefully considered design plan. Entire Net-Work is often installed after a major local or wide area network is already in place.

Whether you are working with a new network or an existing one, it is important to make and follow a plan when installing Entire Net-Work in order to best utilize the existing facilities and provide the best possible response time to your applications.

## Entire Net-Work Components

---

The information in this section refers to the following Entire Net-Work components:

node	An Entire Net-Work instance.
target	A source of information, such as Adabas, EntireX Communicator, or Entire System Server.
client	An application that is accessing a target.

## Topologies

---

Entire Net-Work should be designed to use the same topology used by the underlying network. There are usually several underlying layers of logical and physical configurations that are invisible to Entire Net-Work. Those who design and maintain these underlying layers should be involved in the network design process.

Several topologies are discussed in this section. The bus topology is not included because it does not readily apply to Entire Net-Work configurations.

- [Point-to-Point](#)
- [Star](#)

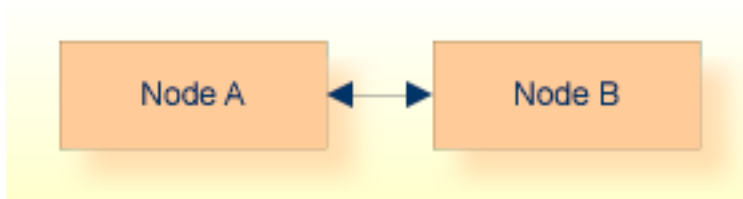
- Ring

### Point-to-Point

Point-to-point is the simplest of all configurations. It consists of two nodes connected by a link, and is the way most networks start; for example, a workstation client application using a direct connection to access a mainframe database.

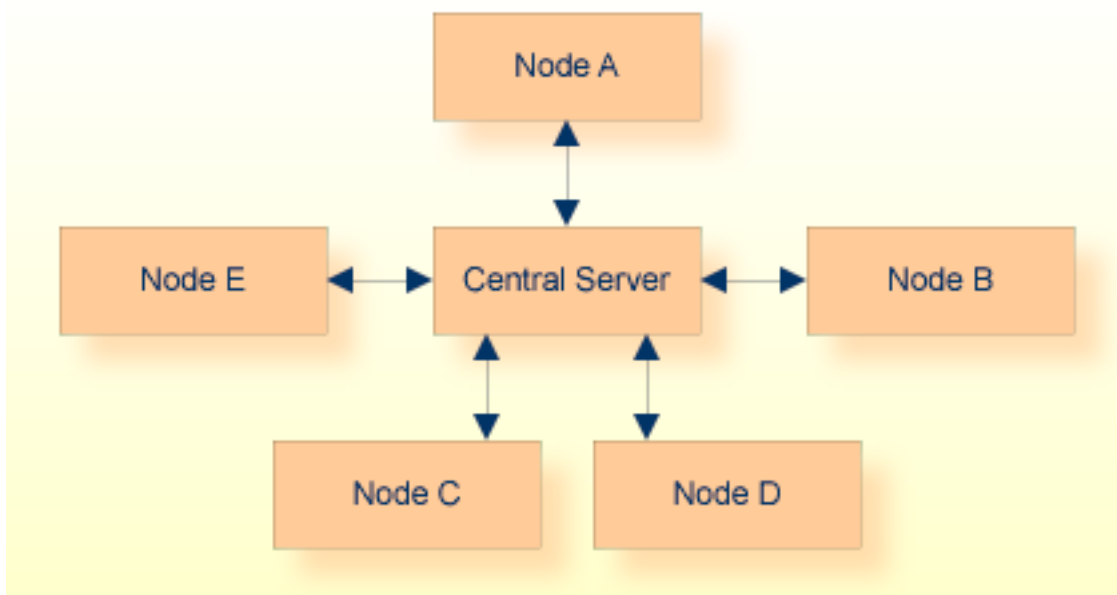
Many modern networks run TCP/IP in a formation that logically consists of many point-to-point sessions. For example, a given client could be attached to `FTP://157.189.1.1/etc/file` as its server.

The following diagram illustrates the point-to-point configuration:



### Star

Point-to-point networks often evolve into Star networks. A central controlling node is attached to each node in the network. The following diagram illustrates the star configuration:



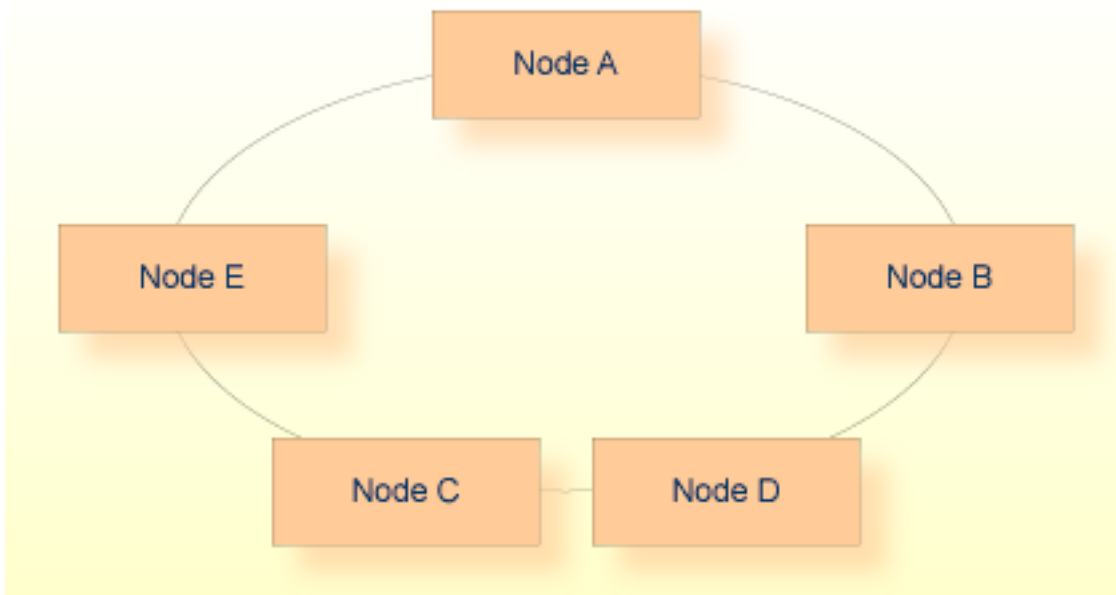
Although targets are normally run on the Central Server, a target running on any attached node is only two hops away from any potential client. The Star configuration is limited by the speed of

the central processor. Its advantage is that it allows central control of the network, which facilitates problem determination.

The original IBM VTAM networks were designed as Star networks, where VTAM itself ran in the central processor and had absolute control over all attached physical and logical units.

## Ring

As its name implies, nodes in a Ring network are arranged in a ring, as shown in the following diagram:

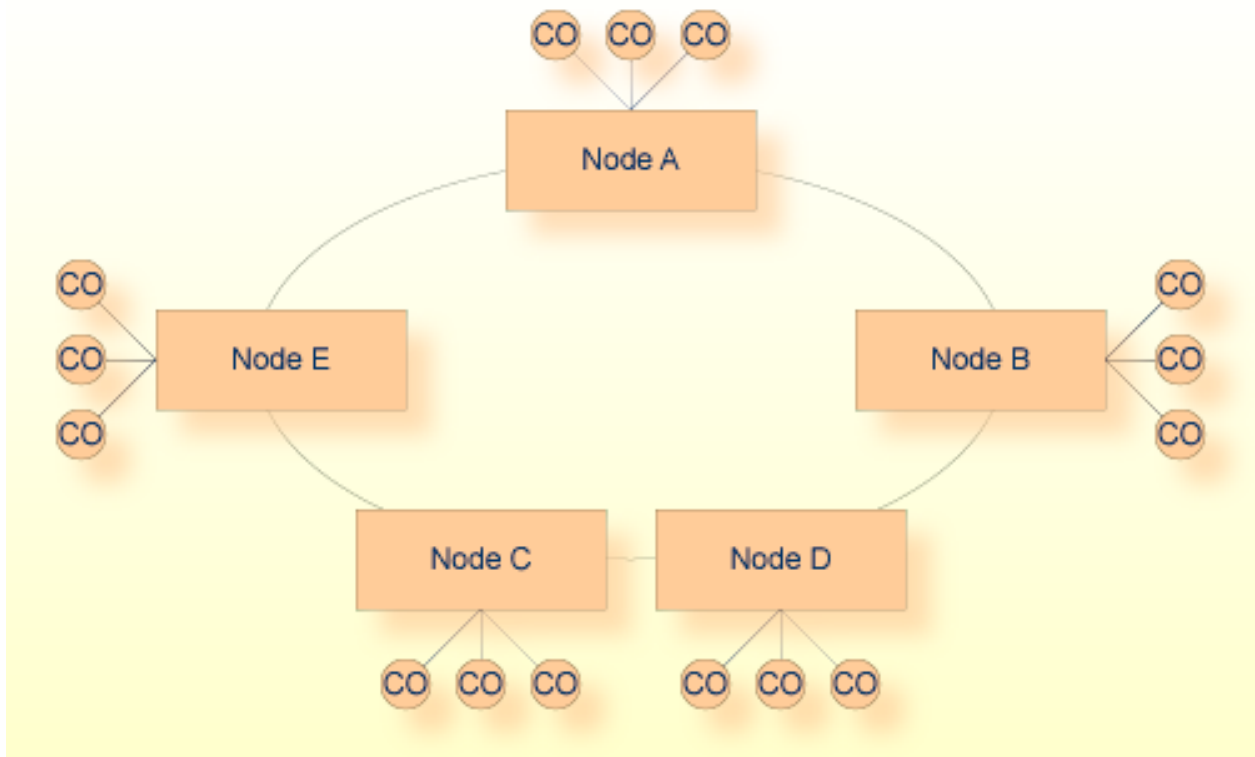


In the Ring configuration, it is not apparent which node is the server and which is the client. For this reason, the Ring network is often used among peers, such as Unix servers or mainframes, where clients or servers may run on any node. Because all nodes are peers, problem determination can sometimes be difficult. An advantage of the Ring configuration shown above is that any node is only two hops away from any other node. This configuration also provides **built-in redundancy**.

## Expanding the Network Configuration

---

It is difficult to determine the best way to build your Entire Net-Work configuration. The Star configuration tends to work best in smaller networks, allowing some central control and the ability to route traffic in the most advantageous way. As the network grows, it is important to carefully consider the placement of targets within the network to keep them logically or physically closer to the clients they serve. This may require a combination of topologies. The configuration in the following diagram, for example, contains a ring of Star networks:

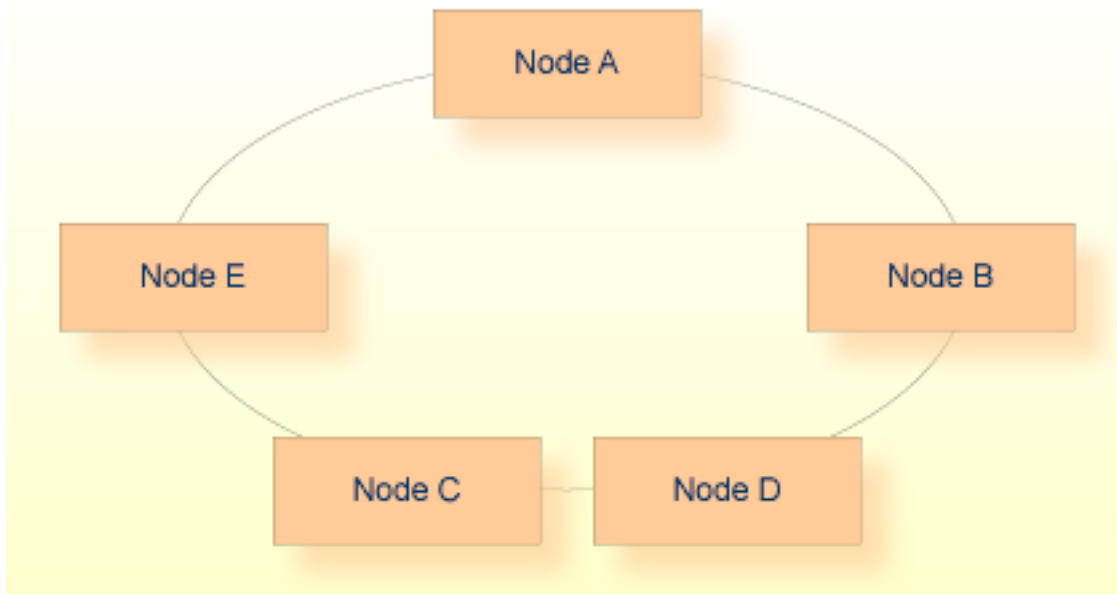


Each server node (Node A, Node B, etc.) acts as the central processor for a group of client-only (CO) nodes, and the server nodes are arranged in a ring formation. None of the client-only nodes is allowed to run a target, so no target can be more than four hops away from any client.

## Redundancy

For purposes of this discussion, redundancy means providing multiple paths between nodes. The underlying network may already be redundant. Consult those responsible for maintaining it before implementing redundancy with Entire Net-Work. If implemented incorrectly, the higher level redundancy could defeat the underlying redundancy. Redundancy can be implemented with Entire Net-Work by running heterogeneous links in parallel or by providing multiple paths between nodes via interconnectivity.

Interconnectivity requires that a physical path exist between all points. In the following diagram, for example, traffic can pass in either direction around the ring. If the link between node C and node D is severed, traffic must pass through nodes B, A, and E to connect node D to node C.



When constructing a logical ring in order to provide redundancy, it is best to determine where the majority of the traffic exists. If most of the traffic moves between node A and node E, for example, putting node D in the middle would add unnecessary processing to the traffic pattern.

## Weighting

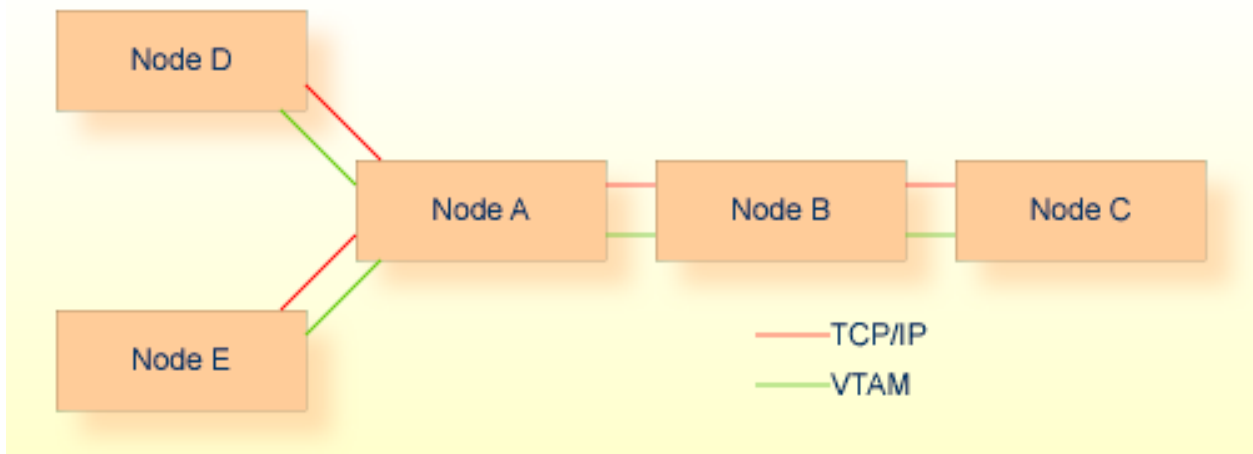
---

Weighting is used to prioritize connections. When two or more connections can be used to reach a target node, Entire Net-Work uses the lowest weighted connection available.

Carefully consider weighting the links in your network to provide the most efficient movement of traffic. Assign the lowest weight to the primary link. Assign higher weights to connections that are slower or more expensive; these connections may be used as backup. If the primary connection is severed, traffic can flow over the backup connection. It is important that the connections not share the same physical path.

The example network in the following diagram has both a TCP/IP connection and a VTAM connection between each pair of nodes. If traffic moves at 10 megabits per second on the TCP/IP link and 56,000 bits per second on the VTAM link, the VTAM link should be used as a backup to the TCP/IP link. Giving the TCP/IP link the lower weight ensures the most efficient flow of traffic.





## Traffic Considerations

The movement of traffic is the core consideration of network design. The following considerations should be kept in mind when designing an Entire Net-Work Configuration:

- Nodes that share the greatest amount of traffic should be directly connected.
- The lowest weights should be assigned to the links with the highest throughput.
- Maintenance traffic should be taken into account.

## Broadcasting

With the exception of client-only nodes, all nodes know about all other nodes in the network and all nodes know about all targets in the network. This is accomplished using broadcast technology.

The first time a node connects to another node in the network, the node that receives the connection sends a message to all the nodes to which it is connected informing them of the availability of the new node. If the new node has others connected to it, then this information is broadcast throughout the network. The newly connected node may also have to broadcast information regarding connected nodes at the other end.

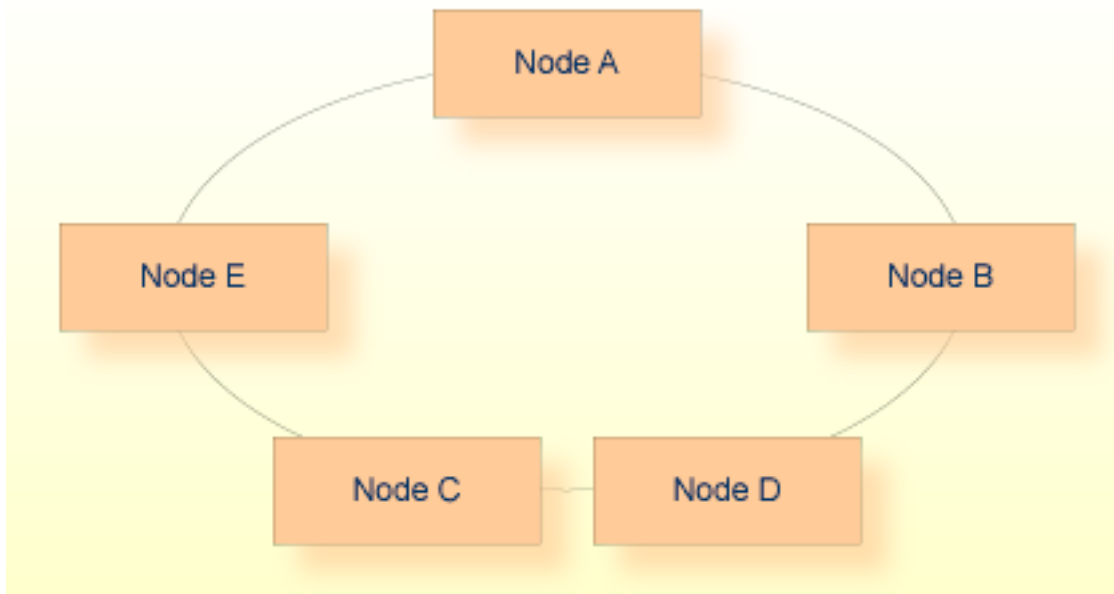
Broadcasting also applies to targets. When a target becomes available or unavailable, the node where it resides will broadcast this information to all the nodes to which it is connected. The broadcast is passed down the line until all nodes in the network are aware of the target's status.

In a large network, broadcasting can create thousands of messages of ever increasing length just to keep every node informed of the status of all known nodes and targets. To prevent wasting

CPU cycles and bandwidth on redundant or unnecessary information, it is important to keep broadcast traffic to a minimum.

There are two good ways to manage this situation:

- Use client-only nodes with the client mode setting turned on.
- Implement redundancy and connectivity in a way that minimizes unnecessary broadcast traffic. Consider the following ring configuration:



In the Ring configuration shown above, it may be tempting to connect node A to nodes C and D, node B to nodes C and E, and so on to provide the most redundant and direct paths possible. Rather than being helpful, however, this creates a great deal of unnecessary broadcast traffic any time a target's status changes on any given node. For example: When a target becomes available on node A, A sends a broadcast to nodes B, C, D, and E. Each of these nodes then rebroadcasts the information to its connected nodes, which in turn creates more broadcast information.



**Caution:** An overly connected network creates a great deal of undesirable traffic and excessive CPU usage. For this reason, Software AG does not support interconnectivity of more than three nodes.

## Client-Only Nodes

Many workstation Entire Net-Work nodes are used only to run clients. Because client-only nodes never contain a running target, information about them need not be broadcast around the network.

Entire Net-Work for the workstation provides a client mode setting. Client mode tells Entire Net-Work that no information about that node is to be broadcast. If the network has 10 machines that run targets, and 1000 machines that run clients only, the use of the client mode setting clearly results in a large reduction in broadcast traffic.

## Number of Hops


A hop is defined as traffic passing from one node to the next. The [point-to-point configuration](#) shows one hop.

Entire Net-Work must manage information about how to move data between all nodes on the network.

Consider the following example:



In order for node A to get to node E, node A must maintain path information that states that node E is available via node B. It must also, however, maintain path information about all other nodes in the path.

 **Caution:** As a path becomes longer, it becomes increasingly difficult to correctly maintain information about all nodes and paths. For this reason, Software AG does not support more than four hops.



## 4 Licensing Entire Net-Work

---

You must install a valid license file on all mainframe platforms in which your Software AG mainframe product is installed.

Please also ensure that the MLC version used corresponds to your version of Entire Net-Work. Details on the required MLC version are given under *Software AG Mainframe Product Licensing*, in the *Entire Net-Work Installation Guide*

The license file is provided as an XML document (encoded in US-ASCII). This document can be viewed using a browsing tool or text editor on a PC. It can also be viewed on the mainframe using the DISPLAY function of the license utility, LICUTIL, described *Software AG Mainframe Product Licensing*, in the *Entire Net-Work Installation Guide*. The license file contains text that represents the licensing information for your product and an associated digital signature, the license key. Among other things, it also displays Software AG legal notices and environmental information.

Your license file is obtained from your Software AG sales representative. If you should have problems with your license file, please contact your sales representative. Please do not edit the file yourself, as you may invalidate it during your attempt. If the product license is incorrect, insufficient, or not installed, Entire Net-Work terminates. Contact your Software AG sales representative for assistance.

For complete information about the licensing process for Software AG mainframe products, read *Software AG Mainframe Product Licensing*, in the *Entire Net-Work Installation Guide*.



## 5 Starting Entire Net-Work

---

Entire Net-Work is started by running a batch job or as a started task. Sample startup (execution) batch jobs can be found in the Entire Net-Work installation instructions for each platform.

Reusable address space IDs (ASIDs) are supported in z/OS environments. So you can specify the z/OS REUSASID system parameter on the start command for Entire Net-Work. For example:

```
/S NETWORK,REUSASID=YES ↵
```

For more information about the REUSASID system parameter, refer to your z/OS documentation.





# 6

## Entire Net-Work Security (z/OS only)

---

■ SAF-based Security Packages .....	28
■ NETSAF .....	28
■ Adabas SAF Security .....	28

## SAF-based Security Packages

---

The System Authorization Facility (SAF) is used by z/OS and compatible sites to provide rigorous control of the resources available to a user or group of users. Security packages such as RACF, CA-ACF2, and CA-Top Secret allow the system administrator:

- to maintain user identification credentials such as User ID and password; and
- to establish profiles determining the datasets, storage volumes, transactions, and reports available to a user.

The resulting security repository and the infrastructure to administer it represent a significant investment. At the same time, the volume of critical information held by a business is constantly growing, as is the number of users referencing the data. The challenge of controlling these ever-increasing accesses requires a solution that is flexible, easy to implement and, above all, one that safeguards the company's investment.

## NETSAF

---

NETSAF (product code WAF) is a separate product for z/OS environments which provides point of access verification for incoming requests.

Refer to *Entire Net-Work SAF Security Interface (NETSAF)* for more information.

## Adabas SAF Security

---

Adabas SAF Security (product code AAF) is a separate product for z/OS environments which enhances the scope of SAF-based security packages by integrating Adabas and Entire Net-Work resources into the central security repository.

With Entire Net-Work version 6.5 SP2 or above, and in conjunction with Adabas SAF Security, the following security-related facilities are available:

- [Protecting Entire Net-Work Start-up](#)
- [Protecting Entire Net-Work Administration Functions](#)



**Note:** The pre-requisites for providing this protection are:

- Adabas version 8.5 SP2 or above
- Adabas Limited Library (WAL) version 8.5 SP2 or above

- Adabas SAF Security version 8.2 SP2 Patch level 2 or above

### **Protecting Entire Net-Work Start-up**

Entire Net-Work start-up can be protected by defining appropriate resource profiles denoting Target ID and SVC number.

Refer to the *Operations* section of the *Adabas SAF Security* documentation for more information on this topic including the format of the resource names used.

### **Protecting Entire Net-Work Administration Functions**

Entire Net-Work administration functions can be protected ensuring unauthorized use of such functions is not permitted.

For example, a user or group may be allowed to use the `DISPLAY` command to display current information about a network component but disallowed from using the `SET` command to dynamically change parameter settings.

Refer to the *Operations* section of the *Adabas SAF Security* documentation for more information on this topic including the list of applicable administration functions and the format of the resource names used.

This protection is available for operator commands issued from both the console and the Programmable Command Interface (PCI).



# 7

## Entire Net-Work Light

---

■ Overview .....	32
■ Use Cases .....	32
■ Installation .....	33
■ TCPL Driver Statement .....	36
■ TCPL Driver Operator Commands .....	45
■ Entire Net-Work Light Operator Commands .....	48
■ Messages Specific to Entire Net-Work Light .....	49

## Overview

---

Entire Net-Work Light (product code WCT) is a subset of Entire Net-Work (product code WCP) that is optimized for client access to z/OS Adabas Databases via TCP/IP. Entire Net-Work Light uses the Software AG XTS protocol and offers performance improvements over Entire Net-Work when used with direct client configurations.

When started on an SVC, WCT registers all eligible database on that SVC with the ADI. The database information is also updated when the TIMER interval expires or the REFRESH command is issued. In addition, WCT delivers calls from client applications to the relevant databases via the Software AG XTS protocol. WCT makes it so every client application initiates and uses its own unique connection.

WCT supports the Entire Net-Work for zIIP option and a WCT server can co-exist with another Entire Net-Work node (WCP, WCA or WDX) on the same SVC.

Entire Net-Work Light has the following restrictions:

- You can define only the `TCPL DRIVER`.
- Connections to Entire Net-Work nodes (WCP, WCA, WDX or ADATCP) are not supported.
- You cannot define LINKs.
- You cannot establish WCT as an Adabas target.
- The Programmable Command Interface (PCI) is not implemented.
- If you want to access an Adabas database with a Linux or Windows client, the database must be UES enabled. The requirements listed in the System Requirements section of this document also apply, unless an exception is mentioned in this section.

## Use Cases

---

Entire Net-Work Light provides Adabas Database access for all Software AG direct client software that uses the Software AG XTS protocol over TCP/IP. Some of these direct clients are:

## Jopaz

The Software AG Jopaz product allows Java programs to make Adabas calls. If the Adabas Database is not local to the Java program, Jopaz will send the Adabas call via TCP/IP (using the Software AG XTS protocol) to the Entire Net-Work Light that has registered the database in the Adabas Directory Server.

## z/OS TCP/IP Enabled ADALNK

The Software AG batch/TSO Adabas link routine, ADALNK, can be configured to support TCP/IP sockets. When configured, z/OS batch/TSO programs running on a different z/OS LPAR can make Adabas calls to any Adabas Database that has been registered in the Adabas Directory Server by Entire Net-Work Light.

## Entire Net-Work Client (WCL)

The Entire Net-Work Client provides a direct TCP/IP connection to Entire Net-Work Servers for Natural and 3GL applications on Linux, Unix, Windows, including Entire Net-Work Light. The TCP/IP connection is made directly to the Entire Net-Work Light node that is identified in the Adabas Directory entry for the Adabas Database. The direct connection potentially eliminates the need for multiple hops through traditional Entire Net-Work nodes.

## ADATCP Substitute

ADATCP provides direct TCP/IP connections to Adabas Databases by running an Entire Net-Work node as a subtask in the Adabas address space. Entire Net-Work Light provides the same direct connectivity without the need for the Adabas address space subtask overhead. The communication responsibility for all directly connected Adabas Databases (on the same SVC as Entire Net-Work Light) is contained in the Entire Net-Work Light address space.



**Note:** ADATCP can receive connections from other Net-Work nodes. Entire Net-Work Light does not have that functionality.

## Installation



**Caution:** Except for license files, Entire Net-Work Light does not use components from WCP or WTC. Do not mix or merge the WCT load library with the WCP or WTC load libraries.

- [Prerequisites](#)
- [Step 1: Unload the Installation Data Sets](#)
- [Step 2: Prepare the Licenses](#)
- [Step 3: Create the Startup JCL / PROC](#)
- [Step 4: Create the ADARUN control statements](#)

- [Step 5: Create the NODE and DRIVER Control Statements](#)
- [Step 6a: If Needed, Install the Adabas Directory Server \(ADI\)](#)
- [Step 6b: If Needed, Start the Adabas Directory Server \(ADI\)](#)
- [Step 7: Start Entire Net-Work Light](#)

## Prerequisites

Entire Net-Work Light requires WAL854.MVSL001 or higher, which contains the fixes AI854009, AU854017 and AU854022.

The following fixes are required for previous Adabas versions:

ADA853: AI853017, AU853059, AU853064

ADA852: AI852023, AU852075, AU852080

## Step 1: Unload the Installation Data Sets

WCT <i>vrs</i> .LOAD	The z/OS load library for Entire Net-Work Light.
WCT <i>vrs</i> .MVSSRCE	The z/OS source library for Entire Net-Work Light, where <i>vrs</i> represents the version of Entire Net-Work Light.
MLC <i>vrs</i> .MVSJOBS	The sample job library for Software AG's common mainframe license check software.
MLC <i>vrs</i> .MVSLOAD	The load library for Software AG's common mainframe license check software.



**Note:** The current version of each library is represented by the *vrs* replaceable.

## Step 2: Prepare the Licenses

Entire Net-Work Light requires both an Entire Net-Work (WCP) and an Entire Net-Work TCP/IP Option (WTC) license. An AZPAD (Adabas for zIIP) license is also required for Entire Net-Work Light with an ADARUN ZIIP=YES option. Please refer to Software AG Mainframe Product Licensing for more information about how to prepare the necessary licenses.

## Step 3: Create the Startup JCL / PROC

The member JCLNET (from the WCT*vrs*.MVSSRCE installation dataset) contains sample JCL for starting Entire Net-Work Light. Modify the dataset names as required.



#### Step 4: Create the ADARUN control statements

The member ADARUN (from the WCTvrs.MVSSRCE installation dataset) contains sample ADARUN parameters for Entire Net-Work Light. Modify the SVC (and any other) ADARUN parameter as required.



**Note:** ADARUN TARGETID=nnnnn is not required for WCT, unless you are using Adabas SAF Security. Please see [Entire Net-Work Security](#) for additional information.

#### Step 5: Create the NODE and DRIVER Control Statements

The members NETWKL and DRTCPL (from the WCTvrs.MVSSRCE installation dataset) contain sample NODE and DRIVER control statements. Change NODENAME to a node name that is appropriate for your site and set the SERVERID, ADIHOST and ADIPORT DRIVER parameters. ADI=YES is specified in the sample.

SERVERID is the TCP/IP port that Entire Net-Work Light listens on. It must be a port number that is not used by any other server on the z/OS LPAR.

ADIHOST is the host name on which the Adabas Directory Server resides in your network.

ADIPORT is the TCP/IP port number that the Adabas Directory Server listens on. In most cases, Entire Net-Work Light must register databases with an Adabas Directory Server to function properly.

Refer to the Entire Net-Work NODE Statement section for a full description of the NODE parameters.

Refer to the [TCPL DRIVER Statement](#) section for a full description of the TCPL DRIVER parameters.

#### Step 6a: If Needed, Install the Adabas Directory Server (ADI)

You only need one Adabas Directory Server in your network. If you need to install the Adabas Directory Server on z/OS, the required installation datasets come with Entire Net-Work (WCP). Refer to the ADABAS Directory Server Administration section for more information.

#### Step 6b: If Needed, Start the Adabas Directory Server (ADI)

Ensure that the identified in the DRIVER ADIHOST and ADIPORT definition Adabas Directory Server is running. If the Adabas Directory Server was installed during the installation of Entire Net-Work Light, please refer to the ADABAS Directory Server Administration section for more information.

## Step 7: Start Entire Net-Work Light

You can start Entire Net-Work Light either by submitting the Entire Net-Work Light job or by starting the Entire Net-Work Light PROC with the z/OS `START` command.

## TCPL Driver Statement

---

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

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](#)
- [Modifying the DRIVER Statement Parameters](#)
- [DRIVER Statement Parameters](#)

### DRIVER Statement Format

The TCPL DRIVER statement has the following format:

```

DRIVER TCPL [ADI = { Y | N } ]
            [ADIHOST = { hostname } ]
            [ADIPART = { partition name } ]
            [ADIPORT = { port number } ]
            [ALLOWIP6 = { Y | N } ]
            [CONNQUE = { n | 10 } ]
            [DRVCHAR = { character | # } ]
            [DRVNAME = { driver-name | TCPL } ]
            [KEEPALIV = { Y | N } ]
            [MULTSESS = { N | Y } ]
            [NODELAY = { N | Y } ]
            [NUMUSERS = { number | 100 } ]
            [OUTBUFSZ = { buffer size | 32k } ]
            [RCVBUFSZ = { buffer size | 32k } ]
            [SERVERID = { n | 1996 } ]
            [SUBSYS = { subsys-name | VMCF } ]
            [SUPMSGs = { Y | N } ]
            [TRACE = { Y | N } ]
            [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. They 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 *TCPL 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 TCPL DRIVER statement.

- [ADI Parameter](#)
- [ADIHOST Parameter](#)
- [ADIPART Parameter](#)
- [ADIPORT Parameter](#)
- [ALLOWIP6 Parameter](#)
- [CONNQUE Parameter](#)
- [DRVCHAR Parameter](#)

- DRVNAME Parameter
- KEEPALIV Parameter
- MULTSESS Parameter
- NODELAY Parameter
- NUMUSERS Parameter
- OUTBUFSZ Parameter
- RCVBUFSZ Parameter
- SERVERID Parameter
- SUBSYS Parameter
- SUPMSGs Parameter
- TRACE Parameter
- TRACESIZ Parameter
- USERID Parameter
- WCPPART Parameter

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

#### ADI Parameter

**ADI = { Y | N }**

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

This parameter specifies if Adabas Directory Server (ADI) support is enabled or disabled. Default is "N" for Net-work nodes, and "Y" for ADATCP and WCT.

- If "Y" is specified, Adabas Directory Server (ADI) support is enabled. This is the default for ADATCP and WCT.
- If "N" is specified, Adabas Directory Server (ADI) support is disabled. This is the default for Net-work nodes.

#### ADIHOST Parameter

**ADIHOST = { *hostname* }**

No default.

Valid values are 1-255 characters.

This parameter specifies the hostname of the Adabas Directory Server (ADI). The hostname is used to attempt to acquire the TCP/IP address of the system where the ADI resides. If used, ADIHOST and **ADIPORT** must both be specified.



**Note:** If you specified ADI=Y, but not ADIHOST, Net-Work Light will attempt to resolve the ADI hostname by trying SAGXTSDSMF and LOCALHOST, in that order.

#### ADIPART Parameter

**ADIPART = {partition name}**

No default.

Valid values are 1-32 characters.

This optional parameter specifies the partition name to be used with the Adabas Directory Server (ADI). If specified, the partition name will be included in all target entries added to the ADI by this session. Partitions are used to restrict database access; when an application queries the ADI for a target and specifies a partition, only entries with the same partition name are returned. Likewise, if the query does not specify a partition, only entries that do not have a partition are returned.

#### ADIPORT Parameter

**ADIPORT = { port number }**

No default.

Valid values are 1-65535.

This parameter specifies the port number used to communicate with the Adabas Directory Server (ADI). If used, **ADIHOST** and ADIPORT must both be specified.



**Note:** If you specified ADI=Y, but not ADIPORT, Net-Work Light will attempt the connection using either the ports SAGXTSDSMFPORT or 4952. If defined, SAGXTSDSMFPORT resolves to an encoded port number. Please see *Adabas Directory Server Concepts* for more information.

**ALLOWIP6 Parameter**

**ALLOWIP6 = { *Y* | N }**

This optional parameter determines whether the line driver will accept connections using IPv6 communication. When the driver is opened, initialization for IPv6 communication is attempted. If the stack is not IPv6-enabled, IPv4 communication is used.

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

- If "Y" is specified, Entire Net-Work will attempt IPv6 communications when the driver is opened.
- If "N" is specified, Entire Net-Work will not attempt IPv6 communications when the driver is opened.

**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 | TCPL }
```

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 "TCPL".

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**

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

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 the TCP/IP 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".

**MULTSESS Parameter**

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

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".

#### **NODELAY Parameter**

**NODELAY = { N | Y }**

This optional parameter allows you to indicate whether the IBM socket option TCP\_NODELAY is enabled or disabled for a link. TCP\_NODELAY indicates whether data sent over the socket is subject to the Nagle algorithm (RFC 896). For more information, refer to your IBM documentation.

Valid values for this parameter are "Y" (the TCP\_NODELAY option is enabled) or "N" (the TCP\_NODELAY option is disabled). The default 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.

#### **OUTBUFSZ Parameter**

**OUTBUFSZ** = *buffer size*

OUTBUFSZ specifies the size of the output buffer.

An output buffer is allocated for each link and is used for the output buffers for an Adabas call, such as the Record Buffer. This preallocated buffer avoids the overhead of allocating and freeing a buffer for each Adabas call. If the output buffer is too small, a separate buffer is allocated for that call only. The link statistics show how many times, if any, the output buffer was too small. The default size is 32K. OUTBUFSZ can be specified in bytes, kilobytes, or megabytes. Example: OUTBUFSZ=2M



## RCVBUFSZ Parameter

**RCVBUFSZ** = *buffer size*

RCVBUFSZ specifies the size of the TCP/IP receive buffer.

A receive buffer is allocated for each link. If the message is too big to fit in the receive buffer, a separate buffer is allocated for that call only. The link statistics show how many times, if any, the receive buffer was too small. The default size is 32K. RCVBUFSZ can be specified in bytes, kilobytes, or megabytes. Example: RCVBUFSZ=64K

## SERVERID Parameter

**SERVERID** = { *n* | 1996 }

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.

The default for this parameter is 1996.

## SUBSYS Parameter



**Note:** This parameter is obsolete. It is available for compatibility reasons only.

**SUBSYS** = { *name* | VMCF }

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.

This parameter is not used in a z/VSE or BS2000 environment.

**SUPMSGSGS Parameter**

**SUPMSGSGS = { Y | N }**

This optional parameter suppresses printout of the following messages in the TCPL DRIVER: NETP818I and NETP819I. This allows you to avoid cluttering your logs with these connection and disconnection messages.

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

- If "Y" is specified, the NETP0818I Connect, and NETP0819I Disconnect messages are suppressed (no longer output to the log).
- If "N" is specified, the NETP0818I and NETP0819I messages are output to the log.

**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`).

**TRACESIZ Parameter**

**TRACESIZ = { size | 4096 }**

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.

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

[ **USERID** = { *userid* | **TCPIP** } ]

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.

## WCPPART Parameter

WCPPART is an alias of **ADIPART** and works in the same way as ADIPART.

## TCPL Driver Operator Commands

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

- [Operator Command Syntax](#)
- [Examples](#)
- [Driver Commands](#)
- [Link Commands](#)

### Operator Command Syntax

Under z/OS, the TCPL Driver operator commands have the following format:

**F NETWORK, TCPL** *target cmd*

The following table describes this syntax.

Syntax Representation	Description
TCPL	Informs Entire Net-Work that the command is destined for the TCPL Driver. If more than one <b>TCPL DRIVER statement</b> exists, use the name specified on the DRVNAME parameter of the <b>DRIVER statement</b> instead of TCPL.
<i>target</i>	A value that informs Entire Net-Work what the target of the command is, as follows: <ul style="list-style-type: none"> <li>■ Specify an asterisk (*) if the target is all links.</li> </ul>

Syntax Representation	Description
	<ul style="list-style-type: none"> <li>Specify the DRVCHAR value ("#" is the default) if the target is the driver itself (see the DRVCHAR parameter on the <a href="#">TCPL DRIVER Statement</a>).</li> <li>Specify the link name if the target is a specific link.</li> </ul>
<i>cmd1, cmd2, and cmdx</i>	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 TCPL Driver operator commands:

```
F NETWORK, TCPL * CLOSE
```

```
TCPL # STATS
```

## Driver Commands

The Entire Net-Work TCPL 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
<u>ALTER</u> <i>driver-parms</i>	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:  TCPL # ALTER ACCEPTUI=Y
<u>CLOSE</u>	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.
<u>OPEN</u>	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.
<u>RESET</u>	Resets all statistics for the driver. Statistics are printed only if the STATS command precedes the RESET command.
<u>SHOW</u>	Displays the current configuration of the driver. The current configuration is always shown automatically following an ALTER command.
<u>SNAP</u>	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.

Command	Action
<u>STATS</u>	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 <code>RESET</code> immediately after the <code>STATS</code> command. For example:  TCPL # STATS RESET
<u>STATUS</u>	Displays the current status of the driver as well as a count of messages received and sent.
<u>TRACE</u>	Causes the TCPL Driver to format and print the driver-specific trace table. The trace table is formatted and printed in hexadecimal automatically when the <code>SNAP</code> command is processed.
<u>USERS</u>	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 TCPL 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.



**Note:** There is rarely a reason to manipulate a link with operator commands because links in Net-Work Light are created dynamically when clients connect and cannot be defined.

Command	Action
<u>CLOSE</u>	Disconnects the link if it is connected to another node and releases all resources held by the link.
<u>CONNECT</u> ↔	Attempts to establish one or more TCPL sessions with the target link(s). If the link is already connected or is in the process of connecting, the command is ignored.
<u>DISCONNECT</u>	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.
<u>OPEN</u>	Allocates all the resources needed by the link to communicate with TCPL. Does not initiate a connect to the remote node. The status of the link displayed via the <code>SHOW</code> operator command is not affected by the <code>OPEN</code> request.
<u>RESET</u>	Resets all statistics for the link. Statistics are printed only if the <code>STATS</code> command precedes the <code>RESET</code> command.
<u>RESUME</u>	Restarts processing on a link that was temporarily stopped due to a <code>SUSPEND</code> command.
<u>SHOW</u>	Displays the current configuration of the link. The current configuration is always shown automatically following an <code>ALTER</code> command.

Command	Action
<u>S</u> NAP	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.
<u>S</u> TATS	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:  TCPL LINK1 STATS RESET
<u>S</u> TATUS	Displays the current status of the link as well as a count of messages received and sent.
<u>S</u> SPEND	Temporarily stops all processing on a link. Processing can be restarted with the RESUME command.
<u>T</u> RACE	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.
<u>U</u> SERS	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.

## Entire Net-Work Light Operator Commands

---

All WCP operator commands are available with WCT with the exception of:

- DEFINE
- DISABLE
- DISPLAY (NODES, PATHS, CSCI, CQ and CQE)
- DISCONNECT
- ENABLE
- PROBE
- RESUME
- SUSPEND

## Messages Specific to Entire Net-Work Light

<b>NET0400I</b>	<b>Net-Work running in WCT mode</b>
<b>Explanation</b>	The Net-Work server is running in WCT mode. Please refer to the <a href="#">Net-Work Light (Product Code WCT)</a> section for details.
<b>Action</b>	No action required, this is an informational message.
<b>NET0401</b>	<b>Driver XXXX not applicable with WCT</b>
<b>Explanation</b>	The indicated DRIVER is not available with WCT. Only the TCPL driver is supported. Please refer to the <a href="#">Net-Work Light (Product Code WCT)</a> section for details.
<b>Action</b>	Remove the DRIVER definition from the DKARTE parameter input.
<b>NET0402</b>	<b>LINK definitions not applicable with WCT</b>
<b>Explanation</b>	The indicated LINK is not available with WCT. No LINK definitions are allowed with WCT. Please refer to the <a href="#">Net-Work Light (Product Code WCT)</a> section for details.
<b>Action</b>	Remove the LINK definition from the DKARTE parameter input.
<b>NET0403</b>	<b>Keyword: XXXXXXXXX not applicable with WCT</b>
<b>Explanation</b>	The indicated keyword used in a NODE or DRIVER definition is not available with WCT. NODE keywords BUFFERS, DEFINE, DOMAIN, MAXPATH, NID0 and ULINK may not be used with WCT. Please refer to the <a href="#">Net-Work Light (Product Code WCT)</a> section for details.
<b>Action</b>	Remove the keyword definition from the DKARTE parameter input.
<b>NET0404E</b>	<b>Operator command not applicable to WCT</b>
<b>Explanation</b>	The following operator commands are not applicable to Net-Work running in WCT mode: DEFINE, DISABLE, DISPLAY (NODES, PATHS, CSCI, CQ and CQE), DISCONNECT, ENABLE, PROBE, RESUME and SUSPEND.
<b>Action</b>	Exchange the operator command with an applicable one.

