Unicenter® SOLVE:CPT™

Administrator Guide

r6.1 SP2

Contents

Chapter 3: CICS/TS Resource Definition Reference

Chapter 4: CICS Sockets Compatibility

Chapter 5: The CPTMRO Environment

Chapter 7: Diagnostic Commands

Chapter 8: Troubleshooting

Chapter 9: Installation Verification Procedure (IVP)

Chapter 10: Translation Tables

Appendix D: The Terminal Administrator Interface

Index

Chapter

1

Introduction

This chapter provides you with an overview of Unicenter® SOLVE:CPT™ .

It discusses the following topics:

- The Unicenter NetMaster Interface—Describes the interface and integration of CICS TCP sockets information into Unicenter NetMaster for TCP/IP management suite
- The Listen Tool—Briefly describes the listen tool provided with Unicenter SOLVE:CPT for establishing TCP/IP connections
- Tracing—Briefly describes the tracing facilities available for application programs in CICS to communicate with a TCP/IP network
- The Unicenter SOLVE:CPT Environment—Briefly describes the installation and customization that must be done to make Unicenter SOLVE:CPT available

The Unicenter NetMaster Interface

Unicenter® NetMaster™ for TCP/IP (Unicenter NetMaster) acts as a central repository for network information in an OS/390 enterprise environment. By enabling this interface, all Unicenter SOLVE:CPT endpoints for a sysplex can be managed from a single point.

Once the Unicenter NetMaster Interface is enabled, Unicenter SOLVE:CPT does the following:

- Passes additional CICS/TS information about TCP endpoints to Unicenter NetMaster such as User ID, CICS/TS transaction name, and CICS/TS transaction number.
- All EZASOKET and EZACICAL calls are time stamped at entry and exit. You can tell whether a transaction is executing inside CICS or API code. You can tell how long a long a call has been active.
- EZASOKET and EZACICAL call statistics are kept at both the session and server level. These session and server level statistics are available for query.
- This information then becomes available via central network management displays within Unicenter NetMaster, refer to the Unicenter NetMaster documentation for additional details.
- Provides an interface, the Unicenter NetMaster command processor allows further drill down inquiries into Unicenter SOLVE:CPT endpoints.

  Note: Unicenter NetMaster 6.2 requires:

  - Service pack 3 (GL0206) installed
  - Place statement 'PROD=SOCKETMGMT' in the region RUNSYSIN file to enable the Unicenter SOLVE:CPT Interface

Unicenter NetMaster Interface Command Processor

The Unicenter NetMaster Interface Command Processor is enabled when you code something on the parameter PORT= in the T09MCMDS macro in the configuration file T09CONez. See the "Configuration Statement Reference" chapter for full details.

Note: It is strongly recommended you turn this feature on by coding something in the T09MCMDS macro.

The Command Processor, a socket-based CICS application, facilitates *administrative connections* between Unicenter NetMaster and Unicenter SOLVE:CPT for CICS. These connections monitor Unicenter SOLVE:CPT EZASocket endpoint activity and initiate other CICS/TS services.

Unicenter NetMaster clients who successfully connect to the Unicenter SOLVE:CPT command processor can have the following commands issued for them to a CICS/TS Unicenter SOLVE:CPT region.

Note: In most cases you are not aware of which command is being issued for you. From the Unicenter NetMaster perspective, all you see is the drill down via a GUI or menu driven screen. The reason these commands are pointed out is that a reference to the command may occur in logging if a problem occurs. In the rare case that you drill down to issuing line mode commands in Unicenter NetMaster, you will see the following list of the supported commands:

CONNDrop	Drop a CPT connection or listener.
CONNGet	Display detailed information about an individual CPT connection.
CONNList	Display a list of CPT connections.
CPTBounce	Restart the CPT interface.
CPTStatus	Display the overall status of CPT.
SMSRVRBounce	Restart the command server (T09TCMDS).
StartServer	Start a CICS listener/server.
StartTransaction	Start a CICS transaction.

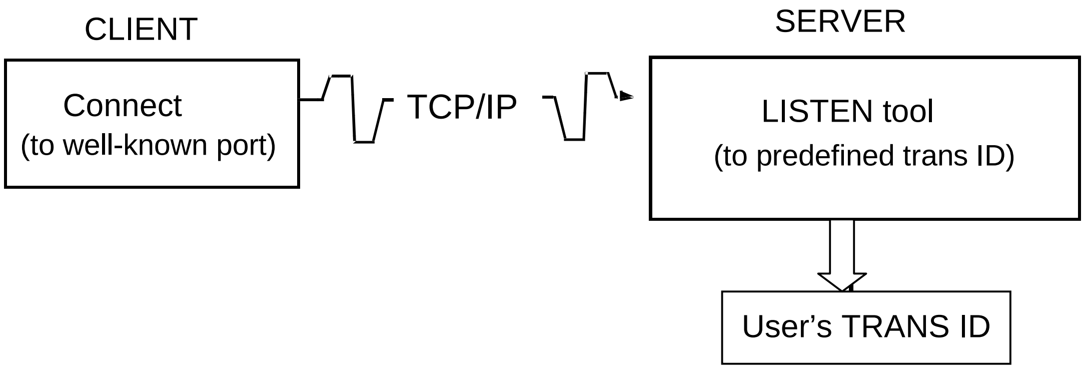
The command processor's well-known port number is transmitted to the clients and can then request connections to the command server across that port. Once a client is verified (assuming command server security was not disabled), you can then drilldown and use menu options that issue the above commands. This enables you to monitor or disable socket connections in the command server CICS region, or issue commands to start other CICS transactions and servers.

The Listen Tool

The Unicenter SOLVE:CPT Listen tool is comprised of prewritten CICS transactions that help establish connections over Transmission Control Protocol (TCP). You can add a new listener service in your environment using two to three parameters of a macro definition.

The T09MLSN macro starts the IPTL server transaction, which replaces the CSKL listener transaction.

Note: There is no limit to the number of listen tools you can start. To customize another listen tool just code another T09MLSTN macro in the T09CONEZ configuration table.



Tracing

The Unicenter SOLVE:CPT API and tools provide extensive debugging and performance analysis capabilities through its tracing and statistics options. The debugging capabilities can be invaluable in an environment where problems can occur in the CICS/TS application, such as the TCPIP address space, the network, or the remote host. The tracing options used selectively can pinpoint exactly where a problem is occurring.

You can follow EZASOKET and EZASOKET calls and their parameter lists both into and out of the TCP/IP region.

You can follow TCP/IP sessions as their ownership passes between CICS tasks.

All trace entries use time stamps that detail events down to the micro second level.

Tracing is turned on and off dynamically—invoked via the TCPEEP interface. There is no need to modify CICS transactions or configuration files. Just submit a TCPEEP job with the options you want traced and the trace events are routed to the trace address space.

All trace output is routed to a trace address space. There is no CICS overhead when formatting trace output. Tracing does not overload the CICS message logs.

The Unicenter SOLVE:CPT Environment

Installation and customization must be performed before Unicenter SOLVE:CPT is available. Installation consists of allocating the Unicenter SOLVE:CPT data sets and executing SMP/E. Customization requires a configuration file and CICS/TS resource definitions.

The configuration file defines the operating environment and automated Unicenter SOLVE:CPT tools transactions. The environment information enables Unicenter SOLVE:CPT to communicate with both CICS and the transport provider’s API. The automated tools transactions are optional.

CICS/TS resource definitions define programs, transaction IDs, and transient data queues. Resource definition statements for Unicenter SOLVE:CPT operating environment programs and transactions are required; while those for Unicenter SOLVE:CPT tools, transient data queues are optional.

Unicenter SOLVE:CPT can be enabled and disabled without shutting down CICS/TS.

You can also:

- Apply maintenance
- Load new configurations, or the transport provider can terminate and reinitialize without CICS/TS being recycled

Chapter 2

Configuration Reference

This chapter provides an introduction to the configuration macros, followed by a detailed reference to assist you in customizing the macros for your environment.

The configuration table contains the information about the Unicenter SOLVE:CPT environment and the automated tools:

- The environmental macro instructions are required.
- The automated tool macro instructions are optional.

This chapter is a reference for the configuration table macros and is **not** an installation guide. The steps required to install these definitions are documented in *Getting Started.*

Note: Other customization information is in the chapters "CICS/TS Resource Definition Reference" and "The CPTMRO Environment."

T09CON*xx* Macros

The following information details the Unicenter SOLVE:CPT required and optional T09CON*xx* configuration macros.

Note: Throughout the guide, the names of the Unicenter SOLVE:CPT configuration file members begin with six characters "T09CON" followed by a two-character suffix. The name of the default configuration file is T09CONFG. The guide uses T09CON*xx* to refer to any valid configuration file name. A valid configuration file name follows all the rules for a valid PDS member name. The use of T09CONcp, T09CONez, and T09CONmr refers to actual sample configuration files in the T09SAMP library. The lowercase two-character suffix of a configuration file name is actually uppercase in the T09SAMP library.

Required T09CON*xx* Configuration Macros

The following macros are required:

T09MCICS

Defines the following:

- Communication subsystem
- Support transaction IDs
- Default translation table
- Log transient data queues to the CICS/TS interface routines

It **must** be the first macro instruction in the T09CON*xx* configuration file.

T09MEND

Defines the end of the configuration file. It does not have any parameter settings; it is an environmental configuration macro instruction.

Optional T09CON*xx* Configuration Macros

The following macros are optional:

T09MLSTN	Defines a LISTEN tool. Each TCP server port can be defined, along with its associated data processing transaction ID. The transaction ID can be the RECEIVE tool or a user-written program. You can define the LISTEN tool to enable the client to determine the transaction ID to start or the transient data queue in which to write client-specified data. You can also define a CSKL replacement listener.
T09MRECV	Defines a RECEIVE tool. The RECEIVE tool provides a simplex data transfer mechanism for receiving network data. Data received from the network is parsed according to configured options and then written to a transient data queue. The transient data queue name can be configured or dynamically resolved.
T09MSEND	Defines a SEND tool. The SEND tool or client facility provides a simplex data transfer mechanism for transmitting network data. Data is read from a transient data queue and parsed according to configured options before it is transmitted over the network. Additionally, static or dynamic server resolution can be configured.
T09MSLCT	Defines the SELECT tool. The SELECT tool or pseudo-conversational tool enables transactions using the RECEIVE API to be pseudo-conversational and not long running tasks.
T09MTRAN	Starts user server transactions at both product startup and while the product is running. It can be used to start any transaction—it does not necessarily have to be a server.
T09MCMDS	Defines the Unicenter NetMaster interface for passing commands from Unicenter NetMaster to Unicenter SOLVE:CPT.
T09MANAG	Defines the session management related parameters.

Assembling the T09CON*xx* Configuration Member

These macro statements can be assembled and link edited into the T09CONCP load module, and placed in the Unicenter SOLVE:CPT load library.

- The T09CONCP member in the T09SAMP library contains the distributed configuration table.
- The T09ASMLK member in the T09SAMP library contains the JCL statements to assemble and link edit the configuration table.

Note: The last two characters of the configuration file name can be any suffix. See the Configuration Suffixing section in the chapter "CICS/TS Resource Definition Reference" for instructions on how to accomplish this.

Macros

This section describes the macros and the parameters used to define the Unicenter SOLVE:CPT environment and tools.

T09MCICS Macro

The T09MCICS macro defines the associated TCP communication subsystem, the CICS/TS transaction IDs, and queue names used by the Unicenter SOLVE:CPT interface.

Important! *There must be only one of these macros in the configuration, and it must be the first statement.*

T09MCICS Syntax

```
T09MCICS
   [  AIPREFIX = ccc ]
   [, AISTATOP = YES | NO ]
   [, AITABSZ = 1000 ]
   [, AITDSTAT = dest ]
   [, AITSELOG = num_storage_recs ]
   [, AITSTRAC = num_temp_stor_recs ]
   [, APIMODNM = EZASOH03 | T02PHPNS ]
   [, CICSENT = Y | N ]
   [, CPT = Y | N ]
   [, EZATRUE = Y | N | C]
   [, JOBNAME = tcpstack ]
   [, LCAFTRAN = IPLF ]
   [, LCASIZE = 4072 ]
   [, LINGER = seconds ]
   [, MSOCK = max_concurrent_sockets ]
   [, MTAKE = seconds ]
   [, QLSTN = backlog_queue_num ]
   [, QNAMES = ( stat, trace, error ) ]
   [, RETRYINT = seconds ]
   [, RETRYMAX = num_retries ]
   [, RSTMSG = Y | N ]
   [, SCTYEXIT= ] NORMAL | MANDTORY ]
   [, SNDTIME = 15 ]
   [, SYNC ]
   [, TRANSID = ( stop, [, |stn ] [, strt ] [, lst2 ]
   [, TRANSTBL = tblname ]
   [, TRCSSN = trace address space subsystem ]
   [, USERID = userid ]
   [, USRTRNID = IPUL ]
```

AIPREFIX	(Administrator Interface Transid Prefix). Three-character prefix of all Administrator Interface transactions to avoid conflict with existing transaction IDs. It also defines the temporary storage queue names used by the Administrator Interface. Default: IPA.
AISTATOP	(Administrator Interface Statistics Option). Specifies whether Unicenter SOLVE:CPT should collect cumulative statistics for online display by the Administrator Interface. Default: No.
AITABSZ	(Administrator Interface Table Size). Number of table entries used to hold Unicenter SOLVE:CPT cumulative statistics for online display. Default: 1000.
AITDSTAT	(Administrator Interface Statistics Destination). An optional transient data destination to which statistics are written when Unicenter SOLVE:CPT is shut down or when statistics capture is reset. Default: Null—No queue name defined.

AITSELOG	(Administrator Interface Temporary Storage for Error Log). Number of temporary storage records to hold Error Log entries for online display.

Maximum: 999. Zero disables the Error Log display.

Default: 30.

Note: When this value is nonzero and error messages are logged, one transaction per message (IPAI) is started in CICS.

AITSTRAC	(Administrator Interface Temporary Storage for Trace). Number of temporary storage records to hold online trace entries.

Maximum: 999. Zero disables the online trace.

Note: When this value is nonzero and trace messages are being logged, one transaction per trace entry (IPAI) is started in CICS.

APIMODNM	The API load module to be loaded at startup (can be either T02PHPNS or EZASOH03). CA's version of T02PHPNS has an alias of EZASOH03. IBM's version of the EZACIC01 TRUE is commonly known as the IBM CICS Sockets product.

Most sites should use the default of EZASOH03. However, CICS sites running both the EZACIC01 TRUE over IBM's TCP/IP stack while concurrently accessing Unicenter SOLVE:CPT over the Unicenter TCPaccess TCP/IP stack all within one CICS region, should set APIMODNM=T02PHPNS. The latter case is equivalent to option #4 shown below. In this case Unicenter TCPaccess must use its version of module T02PHPNS to handle the Unicenter SOLVE:CPT calls.

Default: EZASOH03.

See also parameter: EZATRUE

Note: For a complete description of how the parameters APIMODNM and EZATRUE may be used to meet your environment needs, see CICS Sockets Compatibility in the chapter "Initial Minimal Configuration" of the Getting Started Guide.

A high level description of the four possible scenarios for running the EZACIC01 TRUE is as follows:

1. Sites running the IBM TCP/IP stack with CA's EZACIC01 TRUE Exit inside CICS

```
APIMODNM=EZASOH03,    (the default)
EZATRUE=Y,            (the default)
```

2. Sites running the IBM TCP/IP stack with IBM's EZACIC01 TRUE Exit inside CICS
   APIMODNM=EZASOH03, (the default)
   EZATRUE=N,

3. Sites running CA's Unicenter Solve:TCPaccess stack and CA's EZACIC01 TRUE Exit inside CICS
   APIMODNM=EZASOH03, (the default)
   EZATRUE=Y, (the default)

4. Sites running the EZACIC01 TRUE over IBM's TCP/IP stack while CPT runs over CA's Unicenter Solve:TCPaccess TCP/IP Stack
   APIMODNM=T02PHPNS
   EZATRUE=N
   JOBNAME=TCPaccessJobname

CICSENT

Obsolete. It may be specified, but it is not used.

CPT

Enables or disables all Unicenter SOLVE:CPT features.

By enabling Unicenter SOLVE:CPT, you can use the following features:

- Unicenter SOLVE:CPT tools (LISTEN, RECEIVE, and SEND)
- Unicenter SOLVE:CPT API
- Unicenter SOLVE:CPT CPTMRO
- SELECT tool
- FTP client interface
- For Unicenter NetMaster Socket Management for CICS customers, the following features:
  - Unicenter NetMaster centralized network operations management interface
  - EZASOKET/EZACICAL API (IBM CICS compatibility)
  - CSKL compatibility listen tool
  - Extensive tracing capabilities
  - The administrative interface (IPAM transactions)

Y=All	Unicenter SOLVE:CPT and Unicenter NetMaster Socket Management for CICS features are enabled.
N=Only	Unicenter NetMaster Socket Management for CICS features enabled.

Default: Yes.

DNRSSN	Obsolete. It may be specified, but it is not used.

EZATRUE

The EZATRUE configuration option parameter states whether you will enable and run the EZACIC01 TRUE exit.

Unicenter SOLVE:CPT will try to enable a TRUE called EZACIC01 when a site sets EZATRUE=Y. This allows Unicenter NetMaster Sockets Management and Unicenter SOLVE:CPT to run, debug and monitor CPT, EZASOKET, EZASOKSO, and EZACICAL applications over both IBM's and CA's TCP/IP stacks. This is feature of SOLVE:CPT is called the CICS Sockets Compatibility feature.

Any application program in CICS that makes a program call to EZASOKET; EZACICAL; or EZASOKSO uses the EZACIC01 TRUE. This interface is commonly called the EZASOKET API (Application Programming Interface) or simply the CICS Sockets product from IBM.

If you set EZATRUE=N, the EZACIC01 TRUE exit name is available for use by IBM's CICS Socket product. If you disable the CICS Sockets Compatibility feature by setting EZATRUE=N, you will not be able to use the diagnostics capabilities of SOLVE:CPT to debug and monitor all the previously mentioned API type applications, and can only diagnose issues with SOLVE:CPT applications.

If you set EZATRUE=C, the EZACIC01 TRUE checks to see if the EZACIC01 TRUE exit is already running. If the exit is not running, the SOLVE:CPT will attempt to start the appropriate EZACIC01 TRUE exit.

Default: Y.

See also parameter: APIMODNM

Note: For a complete description of how the parameters APIMODNM and EZATRUE may be used to meet your environment needs, see CICS Sockets Compatibility in the chapter "Initial Minimal Configuration" of the Getting Started Guide.

A high level description of the four possible scenarios for running the EZACIC01 TRUE is as follows:

1. Sites running the IBM TCP/IP stack with CA's EZACIC01 TRUE Exit inside CICS
```
   APIMODNM=EZASOH03,    (the default)
   EZATRUE=Y,            (the default)
```

2. Sites running the IBM TCP/IP stack with IBM's EZACIC01 TRUE Exit inside CICS
   ```
   APIMODNM=EZASOH03,    (the default)
   EZATRUE=N,
   ```

3. Sites running CA's Unicenter Solve:TCPaccess stack and CA's EZACIC01 TRUE Exit inside CICS
   ```
   APIMODNM=EZASOH03,    (the default)
   EZATRUE=Y,            (the default)
   ```

4. Sites running the EZACIC01 TRUE over IBM's TCP/IP stack while CPT runs over CA's Unicenter Solve:TCPaccess TCP/IP Stack
   ```
   APIMODNM=T02PHPNS
   EZATRUE=N
   JOBNAME=TCPaccessJobname
   ```

JOBNAME	Required. Job name or step name of the TCP/IP stack to be used. **Note:** This is the job name of either a IBM TCP/IP stack or a Unicenter TCPaccess stack. Default: TCPIP.
LCAFTRAN	Transaction ID for the program T09TLCAF. It is used to free LCA storage at product shutdown. Default: IPLF.
LCASIZE	Below the line storage allocation for use as a LCA control block. Each server that runs requires 12 bytes of LCA storage. Additionally, there is a 12-byte header for the LCA storage. Valid values: 0 to 32767. Default: 612.
LINGER	Seconds to wait on a TCP orderly close call to complete. This is the reception of a FIN-ACK at the TCP transport layer from the remote host in response to a FIN (finish) initiated from the local connection endpoint. A FIN is created due to a shutdown socket API call. Default: One.
MSGL	Obsolete. It may be specified, but it is not used.

MSOCK	Maximum number of concurrent sockets per initapi socket call that Unicenter SOLVE:CPT supports. Typically, each connection requires a socket. In general, every server and every individual transaction calls initapi or has one implied for them by doing a socket function requiring an initapi call. The MSOCK value on the T09MCICS statement will be used for servers whenever it is larger than the MSOCK value on the T09MLSTN statement. The MSOCK value on the T09MCICS statement will be used for servers whenever it is larger than the ACMMSOCK value on the ACM parameter list. **Note:** Only one initapi call is implied per CICS/TS transaction number. Maximum: 2000. Default: 50.
MTAKE	Maximum number of seconds to wait for a spawned CICS/TS task (transaction) to takesocket a connection. If the time-out occurs before the spawned task does a takesocket(CPT TAKE) call, the connection is closed, only if the TS queue created with the EXEC CICS START FROM option no longer exists. That is, an EXEC CICS RETRIEVE call was done by the new task or the task exited. Default: Zero.
QLSTN	Listen Backlog Queue used on all LISTEN service request calls unless overridden by *setsockopt*. Default: Five.
QNAMES	Transient data queue names for various message classes produced by Unicenter SOLVE:CPT routines. Entries must be defined in the CICS/TS RDO for each of the queue names specified, or Unicenter SOLVE:CPT initialization will fail.

stat	Transient data queue name where statistics information is logged. Default: ACST.
trace	Transient data queue name where trace information is logged. Default: ACTR.
error	Transient data queue name where error messages information is logged. Default: ACER.

RETRYINT	Interval between restart attempts in seconds. Valid values: 5 to 86400. Default: 120 (two minutes).
RETRYMAX	Number of times Unicenter SOLVE:CPT attempts to restart if the TCP provider is not available. Valid values: 0 through 999. A value of zero means that no restart is attempted. Default: Zero.
RSTMSG	The RSTMSG parameter determines whether CPT will report instances of disconnect for a session in the CPT CICS logs. CPT will suppress disconnect related error messages associated with an ERRNO=00000036 inside the SEND and RECEIVE calls whenever a site sets RSTMSG=N. Whenever a site runs with RSTMSG=Y all disconnect events should be reported in the CICS/CPT logs when discovered inside either the SEND or RECEIVE process. Valid values are Y or N. If the first character in the RSTMSG parameter is not N then Y will be used for the RSTMSG parameter. Whenever a site runs with RSTMSG=N it may reduce the site's ability to recognize a new network or application problem. However there are a number of CPT application sites that do not always close their sessions gracefully. So the removal of these messages would significantly reduce the number of repetitive unnecessary messages in the CICS logs. Default: Y.
SCTYEXIT	The name of the security exit program to execute when any listener creates a new connection. For more information, see the Security Program section in the appendix “Security.”

SCTYTYPE	How the security exit program gets control if SCTYEXIT is coded on the T09MLSTN and T09MCICS macros.	
	NORMAL	The security exit program specified in the T09MLSTN macro SCTYEXIT parameter can override what is coded In the T09MCICS SCTYEXIT parameter.
	MANDATORY	The security exit program specified in the SCTYEXIT parameter of the T09MCICS macro is always used. **Note:** Anything specified at the listener T09MLSTN macro is ignored.
	For more information, see the Security Program section of the appendix “Security.” Default: NORMAL.	
SNDTIME	Defines the number of seconds a session will wait for a SEND call to complete sending all data when a SEND call fails with EWOULDBLOCK condition. A SNDTIME of zero means that SEND call would never time out. The EWOULDBLOCK condition could occur for a session when an endpoint has sent so much data that the remote receive window has gone to zero and data has filled up the SEND TCP buffers at the TCP/IP stack level. Valid values: 0 to 14439. Default 15.	
SSN	Obsolete. It may be specified, but it is not used.	
TCP	Obsolete. It may be specified, but it is not used.	
TRANSID	Defines transaction IDs used in Unicenter SOLVE:CPT operation.	
	stop	Transaction ID for the program T09TTERM as defined to the CICS/TS RDO definitions. This transaction ID disables the currently active Unicenter SOLVE:CPT environment. Default: IPPR.
	lstn	Transaction ID for the program T09TLSTN as defined to the CICS/TS RDO definitions. It must be specified if any T09MLSTN macros follow in the configuration. This transaction starts a Unicenter SOLVE:CPT Listen tool. Default: IPTL.

	strt	Transaction ID for the program T09TSTRT as defined to the CICS/TS RDO definitions. This transaction ID enables the Unicenter SOLVE:CPT environment, starts listeners, and starts the Unicenter NetMaster interface. Default: IPST.
	lst2	Transaction ID for the program T09TLST2 as defined to the CICS/TS RDO definitions. This transaction is the second half of the *lstn* transaction and is used when the security or Client-Data option is in effect.

TRANSTBL

Default translation table name for all following configuration statements. For the procedure to set up a custom translation table, see the chapter “Translation Tables.”

tblname	Specifies the name of the default ASCII to EBCDIC and EBCDIC to ASCII translation table for the rest of the configuration. This can be any one of the tables provided with the product.

Default: T09XENG.

TRCSSN

Subsystem identifier for the trace address space subsystem identifier. When the trace address space is started, you can use TCPEEP to trace EZASOKET or EZACICAL calls, and their results.

Default: ACTR.

USERID

User ID that the LISTEN API or LISTEN tool uses when starting child transactions. It lets these child transactions inherit the security permissions of the specified user ID.

When this parameter is specified on the T09MCICS statement, it takes effect for all LISTEN API or LISTEN tools that do not specify a USERID parameter.

USRTRNID

Transaction ID for the program T09TULST. It is used to start a particular T09MTRAN entry in the T09CON*xx* configuration file.

Default: IPUL.

T09MLSTN Macro

The T09MLSTN macro defines a port for a TCP listener in CICS/TS. When a connection is made, a defined data processing (child) transaction is initiated. Optional buffering specifications, statistics, and tracing options can be defined.

Note: You can configure and start any number of T09MLSTN listener macros. However, for each macro coded, you generate one long running transaction in CICS/TS.

IBM TCP/IP only: For every T09MLSTN macro coded in the T09CON*xx* configuration file, you may need to define port security in the PORT section of the profile.tcpip data set as follows:

```
1234 TCP cicsprod
```

Where:

1234	PORT=1234 parameter
cicsprod	Name of your CICS/TS started task

Unicenter TCPaccess running port security: For every T09MLSTN macro coded in the T09CON*xx* configuration file, you may need to define port security in the PORT section of the TCPBND*xx* configuration file.

For example, the following PORT statement reserves TCP port 1234 with associated job CICSPROD in the TCPBND*xx* configuration file:

```
PORT NUM(1234)
     PROTO(TCP)
     JOBN(CICSPROD)
     ACCESS(SHR)
```

T09MLSTN Syntax

```
T09MLSTN
   [, APISTAT = ( [ CONN ] [, TERM ] ) ]
   [  CLNTIME = seconds ]
   [, CLNTLEN = data_len ]
   [, CLNTRNS = YES | NO ]
   [, CLNTTBL = translation_table ]
   [, DNR = YES | NO ]
   [, MSOCK = max_sockets ]
   [, PARM = label ]
   [, PASSDSC = N |Y ]
   [, PORT = ( number ) ]
   [, QLSTN = num_backlog_queue ]
   [, RCVBUF = ( number , size ) ]
   [, SCTYEXIT = exit-program-name ]
   [, SNDBUF = ( number , size ) ]
   [, SOCKCOMP = YES | NO ]
   [, STIMEOUT = P | E ]
   [, TRANSID = transid  ]
   [, USERID = userid ]
```

APISTAT	Defines statistic logging options for the API level code of the Unicenter SOLVE:CPT routines	
	CONN	Specifies that a message be generated upon establishment of either a server or client connection. This message contains protocol address information.
	TERM	Specifies that a statistic message be generated upon termination of a connection. This message contains Task-Related User Exit (TRUE) data transfer information.
APITRAC	TCPEEP tracing has made the APITRAC parameter obsolete. It is left in for backward compatibility. However, it does not effect application execution	
CLNTIME	Client-Data Listener option. Specifies that the LISTEN tool determine which transaction ID to start based on the initial input data stream received on the connection. See Client/Data Listener Option for the required input formats.	
	The CLNTIME value indicates the number of seconds the LISTEN tool waits to receive the input parameter data stream from the client, once the client successfully establishes a connection with the LISTEN tool.	
	Either setting the CLNTIME parameter greater than zero or the TRANSID parameter is required. However, the two parameters are mutually exclusive.	
	Default: Zero (Use of the Client-Data Listener option in not enabled).	
CLNTLEN	Maximum length of data the LISTEN tool tries to receive for the initial data stream.	
	Note: This value is useful when the amount of client data being sent is known and consistent. See Client/Data Listener Option for the required input formats.	
	Default: Initial data stream is a maximum length of 50 bytes.	
	Coding the parameter greatly speeds processing by enabling the LISTEN tool to continue processing new connections without waiting the full CLNTIME value for the initial data.	
	Note: This length includes the entire data stream from the beginning of the transaction name through the actual client data bytes including any imbedded commas.	
CLNTRNS	CLNTTRNS=YES indicates that the initial client data input stream should be translated.	
	Default: CLNTRNS=NO.	

CLNTTBL	Name of the translation table to use for translating the client data input stream. If no translation table is specified, the default translation table is used.
DNR=Y|N	Indicates whether to resolve remote IP addresses into IP names with DNR calls. Default: NO. ***Important!*** *It is strongly recommended that you use DNR=NO, since this can create huge 30 second connection establishment delays if your DNS is not correctly configured to resolve IP names into IP addresses. Most DNS servers do not support this feature, and the call takes 30 seconds to time out. Therefore, your listening port could be in a blocked state, allowing no new connection establishment for a period of 30 seconds while waiting for the failed DNS call to time out.*
MSOCK	Maximum number of concurrent sockets per initapi socket calls the product supports. Typically, each connection requires a socket. In general, every server and every individual transaction calls initapi or has one implied by doing a socket function that requires an initapi. The MSOCK value on the T09MCICS statement will be used for servers whenever it is larger than the MSOCK value on the T09MLSTN statement. **Note:** Only one initapi call is implied per CICS/TS transaction number. Maximum: 2000. Default: 50.
PARM	Defines the assembler language label within the configuration table that points to an automated RECEIVE tool macro (T09MRECV) statement for this server. **Note:** This field is required and valid only when using the RECEIVE tool—it is used to match up listen tools with receive tools due to the fact that the TRANSID parameter can be duplicated in the configuration table when more than one receive tool is used. ■ See Sample Configuration Table with Matching Receive and Send Tools for a picture of how this ties together. ■ See the default IVP configuration sample, T09CONCP, in the T09SAMP library for an example of how the two tools tie together via the PARM parameter.
PASSDSC	Specifies whether a comma can be part of the first data packet passed to the CSKL replacement listener that has been configured by CLNTIME, CLNTLEN, or CLNTRNS on the T09MLSTN statement. PASSDSC=Y permits one or more commas in the first data packet.

	PASSDSC=N causes the application to discard the first data packet whenever a record or subsequent comma is found. Default: N	
PORT	Required.	
	number	Defines the local host's transport provider port number that this server is to listen on.
QLSTN	Listen Backlog Queue. This number defines the number of concurrent connection requests that may be queued up while waiting to process the current connection request. Default: Five.	
RCVBUF	Defines buffering space for the Unicenter SOLVE:CPT API routines. This buffer space is used to retrieve data from TCP/IP. The buffer number and size are multiplied to determine total receive storage allocation.	
	number	You should only specify one. Adding extra buffers wastes storage and does not improve performance. Default: One.
	size	Specifies the maximum size of an input or receive buffer. The combination of these must not exceed 61 KB. The Default: 1024.
SCTYEXIT	Security exit program name to use when this listener processes a connection. The exit may be overriden if the T09MCICS macro has SCTYTYPE=MANDTORY coded. See the Security Program section of the "Security" appendix for more information.	
SERVICE	This field is obsolete. It may be specified but it is not used. **Note:** Sites must use the PORT parameter.	

SNDBUF	Defines buffering space for the Unicenter SOLVE:CPT routines. This buffer space is used to transmit data to TCP/IP. The buffer number and size are multiplied to determine total send storage allocation. The combination of these must not exceed 61 KB.	
	number	You should only specify one. Adding extra buffers wastes storage and does not improve performance. Default: One.
	size	Specifies the maximum size of an output or send buffer. Default: 1024.
SOCKCOMP		
	SOCKCOMP=NO	Specifies this is a CPT/API listener. This type of listener is required for the Unicenter SOLVE:CPT RECEIVE tool and any transaction code using the higher-level UNICENTER SOLVE:CPT sockets API calls as documented in the *Programmers Guide*. **Note:** You must be licensed for Unicenter SOLVE:CPT to use this type of listen tool.
	SOCKCOMP=YES	Specifies this is a sockets compatibility listener. A sockets compatibility listener uses the IBM CICS sockets like calls to the EZASOKET or EZACICAL TCP sockets API modules. This listen tool is completely compatible with IBM CICS sockets CSKL type listener.
	Default: NO.	
STIMEOUT	This parameter states whether daughter server sessions will participate in either the GIVE or session inactivity timeout STEAR process. Setting STIMEOUT=P means daughter sessions from this server will participate in either the GIVE or session inactivity timeout STEAR process. Setting STIMEOUT=E means daughter sessions from this server will be excluded from the STEAR process. There will not be any GIVE or session inactivity timeout for any sessions associated with this server. Default: P (Participate in GIVE and session inactivity timeout).	

SYNC	CICS Syncpoint option. Issue a CICS SYNCPOINT from the LISTEN just before spawning the receive transaction. Default: Null (No Syncpoint).
TRANSID	Defines the transaction to start when a TCP/IP connection is established. Either the TRANSID parameter or a CLNTIME value greater than zero is required. However, the two parameters are mutually exclusive. **Note:** Specifying the PARM keyword is required when using the RECEIVE tool.
USERID	Defines the user ID that this LISTEN tool uses when starting child transactions. This allows the started child transactions to inherit the security permissions of the specified *userid*. **Note:** If this parameter is specified, it takes precedence over any user ID parameter specified on the T09MCICS statement. If security exits are used, they may change the user ID.

Client/Data Listener Option

The Unicenter SOLVE:CPT Client/Data Listener option allows one listening TCP/IP socket port to serve as a multi-function server. This is achieved by passing the CICS/TS transaction name in the initial TCP packet. In this way, a single server can distribute connections to many different applications.

This server is compatible with applications written to use IBM's CICS/TS provided listener CSKL.

Note: This additional server flexibility does have a performance impact. By having the listener receive as part of its processing, the servicing of new connections can be delayed. For this reason, this listen server type is not recommended for high connection volume services.

In an attempt to avoid many of the inherent performance problems, the client/data listener tool service is broken into two transactions:

- The first transaction handles connection establishment thus blocking the port for a minimal amount of time.
- The second phase of the listener waits for the client data independent of blocking the connection establishment port.

To further enhance performance:

- Consider using the CLNTLEN parameter whenever possible
- Start multiple client/data listeners.

That way any high volume applications can be on their own server port independent of low volume applications. There are no restrictions to the number of client data listeners that can be started. By following these suggestions, any possible performance issue can easily be eliminated.

The design of the Client/Data Listener mimics the format of a standard CICS/TS 3270 terminal data stream. The design of the Client/Data Listener mimics the format of a standard CICS/TS 3270 terminal data stream, which is similar to a standard CICS/TS terminal interaction on initialization of a terminal transaction. The first four characters of the initial data packet is the transaction name as if you were connecting from an actual 3270 CICS/TS terminal. Another similarity is that the transaction name can be followed by optional data (parameters) that are passed to the transaction. This is a great listener to have for providing multiple applications with TCP connectivity within just one long running server transaction. Refer to the previous performance notes for other considerations.

The client data/listener works in the following manner:

When a connection is received, the phase two listener is started to unblock the original server listening port.

The phase two listener:

- Does a TCP receive from the network
- Expects one of the following client data formats to be received:

```
TRAN
TRAN,UUUUUUUUUUUU
TRAN,UUUUUUUUUUUU,IC,HHMMSS
TDQN,UUUUUUUUUUUU,TD
TRAN,,IC,HHMMSS
TDQN,,TD
```

Depending on the format of data, the listener determines how the actual spawned application child is started. Continue reading for further details on how this works.

Coding a value in the CLNTTIME field greater than zero turns on the client data listener. There are also options for translating the client data string and changing the translation table if that is desired.

Default: No translation.

TRAN|TDQN	A one- to four-character field followed by an optional comma implying more parameters. The field can contain one of the following: ■ The transaction ID to start ■ A transient data queue (TDQ) name to which the 1 to 35-bytes of optional user data is written—if provided
UUUUUUUUUUUU	1- to 35-bytes of user data that is passed to the started transaction or written to the transient data queue in the field CLNTDATA.
IC	Specifies that transaction TRAN be started in *HHMMSS*. **Note:** If left blank, startup is immediate.
HHMMSS	Hours, minutes, and seconds for the IC option.
TD	Indicates that the optional client data field CLNTDATA(UUUUUUUUU above) will be written into the transient data queue, TDQN.

Client/Data Option Data Structure

The data structure passed to the invoked program has the following format. This structure is accessed by an EXEC CICS RETREIVE command in the invoked (spawned child) transaction.

```
CLNTPARM DS    0F
TOKEN    DS    F     New token – socket ID
         DS    CL16  Reserved
CLNTDATA DS    CL36  Up to 35 bytes of client data
PROTADDR DS    0F
DOMAIN   DS    H     Family
RPORT    DS    H     Remote port
RADDR    DS    F     Remote IPADDR
         DS    XL8'00'  Reserved
CLNTLEN  EQU   *-CLNTPARM
```

Examples

Client/Data Listener with Translation

To invoke the Client/Data Listen Tool and automatically translate the input stream from ASCII to EBCDIC, you must specify the following options in the T09MLSTN parameter:

```
T09MLSTN PORT=2002,CLNTIME=5,CLNTRNS=YES,CLNTTBL=MYTABLE
```

In this example, the Listen tool:

- Listens for connections on port 2002
- Waits for up to five seconds for the input stream after establishing a connection
- Translates the input stream using the translation table MYTABLE

T09MRECV Macro

The T09MRECV macros define the inbound simplex data processing options. When the LISTEN tool establishes a connection the RECEIVE tool is initiated. The data from the connection is placed in the CICS transient data queue as specified by either the QNAME keyword or is resolved dynamically from the first four bytes of the data received. The OPTIONS and SEP keywords define how the data is processed before being written to the specified CICS transient data queue. Typically, these queues have a TRIGLEV= parameter coded for them in the DCT, as well as a TRANSID specified for them in a user-written CICS transaction to process the data. The program name for the RECEIVE transaction is T09TRECV.

T09MRECV Syntax

```
Label   T09MRECV   ,QNAME = name

            [, OPTIONS = ( [ QNAME ] [, TRANS | RAW ] [, LL | SEP | ALL | FILE ] [ STRIP
        | NOSTRIP ] ) ]
            [, SEP = ( CRLF | CR | LF | X'xx..' ) ]
            [, TDQBUF = ( tbuf ) ]
            [, TRANSTBL = tblname ]
            [, TRNSTAT = ( TERM ) ]
```

where:

Label	Required. *Label* is an assembler language label on the T09MRECV macro statement that must match one of the following: ■ The PARM=*label* value on the associated LISTEN tool(T09MLSTN) ■ Or the RECEIVEPARM *label* parameter on the DEFINE SESSION object in the CPTMRO configuration exec See the Sample Configuration Table with Matching Receive and Send Tools or Sample Configurations with Matching Receive and Send Tools Using the MROCPT Feature for a picture of how this ties together. Also, see the default IVP configuration sample, T09CONCP, in the T09SAMP library for an example of how the two tools tie together using the PARM parameter.
QNAME	Keyword defining the transient data queue name for the Unicenter SOLVE:CPT Receive transaction to write its data. The *name* defines the CICS transient data QNAME where the data is sent. **Note:** This keyword cannot be specified if OPTIONS=QNAME is defined, but is otherwise a required parameter.

OPTIONS	Defines how data is processed when it is received on a TCP/IP connection.

QNAME	Specifies that the first record be used to determine the CICS Transient Data QNAME. The first four bytes of the record specify the QNAME and are not written to the queue. **Note:** This option and QNAME=*name* are mutually exclusive.
TRANS or RAW	Specifies whether (TRANS) or not (RAW) to translate the data from ASCII to EBCDIC upon receipt from the remote system.
LL	Specifies that the first two bytes of data is a binary length field that does not include the length field. It has a minimum value of one. That is, there must be one byte of data. The specified amount of data is read from the TCP connection and written to the CICS transient data queue as one record. Maximum length: 32769, for 32767 bytes of data. The length field is not written.
SEP	Specifies that records are separated by characters in the data. The record separators are defined in the SEP=*keyword*. The separator sequence is searched for before the data is translated, if translation is requested.
ALL	Specifies that the entire contents of the CICS transient data queue be received as a single TCP session with no indication of record boundaries.
FILE	Specifies that the entire TCP session be received and written to the CICS transient data queue as one record. This limits the amount of data in one TCP session to 32767 bytes.
STRIP	Specifies that the separator sequence not be written to the transient data queue.
NOSTRIP	Specifies that the separator sequence is written to the transient data queue.

Default: TRANS and SEP.

SEP	Keyword—only applicable if OPTIONS=SEP. SEP= specifies the characters that separate records.	
	CRLF	Specifies that records are separated with the ASCII carriage return-line feed character. The ASCII character for CRLF is x ‘0D0A’.
	CR	Specifies that records are separated with the ASCII carriage return character. The ASCII character for CR is x‘0D’.
	LF	Specifies that records are separated with the ASCII line feed character. The ASCII character for LF is x‘0A’.
	X‘xx..’	Specifies that records are separated with a specified hexadecimal string. The string must be two or four hexadecimal digits long. Data is received from the TCP connection until a separator sequence is found, then data is written to the CICS transient data queue as one record. The maximum length of one record is 32767. The separator sequence is not translated. If the separator sequence appears anywhere within the data, it is interpreted as a record separator by T09TRECV. This may lead to unpredictable results.
TDQBUF	Keyword defining buffering space for the Unicenter SOLVE:CPT routines to build a transient data queue record.	
	tbuf	Specifies the size of the allocated transient data queue buffer. This is the maximum amount of data that can be contained in one logical record. **Note:** A logical record exceeding this size is considered an error and data will be lost.
	Default: 102.	
TRANSTBL	Keyword defining an alternate translation table to use when processing data on connections to this port. Otherwise, the default translation table specified on the T09MCICS statement is used.	
	tblname	Specifies the name of the ASCII and EBCDIC translation table for this entry. For more information on customizing the table, see the chapter “Translation Tables.” **Note:** This parameter overrides the default specified in the T09MCICS statement.

TRNSTAT	Keyword defining statistic logging options for transaction level (automated tool) code of the Unicenter SOLVE:CPT routines.
	TERM — Specifies that a statistic message be generated upon termination of a connection. This message contains transaction data transfer information.
TRNTRAC	TCPEEP tracing has made the TRNTRAC parameter obsolete. It was left in for backward compatibility. However, it will not affect application execution.

Sample Configuration Table with Matching Receive and Send Tools

This is an example of how the Unicenter SOLVE:CPT listen, receive and send tools can be tied together. Below are some excerpts from the default configuration table, T09CONCP that is distributed in T09SAMP library. This particular sample shows the standard Unicenter SOLVE:CPT configuration table setup that is used for the Unicenter SOLVE:CPT IVP. If you are interested in configuring this same environment with the CPTMRO feature, see the diagram on the following page.

Notice the correlation between the two IVPRECV1s and the two IVPRECV2s. The lines and arrows on the left side of the diagram show the correlation between the listen tool (TO9MLSTN) PARM=*label* parameter and the Assembler language label that is on the receive tool (TO9MRECV) definition. This is the required parameter method that Unicenter SOLVE:CPT uses to associate a listen tool and a receive tool together.

The lines and arrows on the right side of the diagram show the correlation between the send tool (TO9MSEND) remote PORT= *parameter* that ties to the local PORT= *parameter* on the listen tool (TO9MLSTN). This is an example of how the send tool is used with the CPT IVP. However, in a production environment, this remote server port on the send tool is normally remote from the MVS CICS environment where you are configuring the send tool. This is different from the receive tool in that this correlation is not required and is only used as an example.

```
.
.
.
         T09MCICS TRANSID=(IPPR,IPTL,IPST,IPT2),             X
               TRCSSN=ACTR,                                  X
               JOBNAME=TCPIP
*
*        Listen tool: Server definitions required
*                     for the IVP to run
*
         T09MLSTN PARM=IVPRECV1,                             X
               PORT=1350,                                    X
               TRANSID=IPTR,                                 X
               APISTAT=(CONN,TERM)
*
         T09MLSTN PARM=IVPRECV2,                             X
               PORT=1351,                                    X
               TRANSID=IPTR,                                 X
               APISTAT=(CONN,TERM)
.
.
.
*
*        Receive tool definitions required for the IVP to run
*
IVPRECV1     T09MRECV QNAME=IPRF,                            X
               OPTIONS=(RAW,FILE),                           X
               TRNSTAT=(TERM)
*
IVPRECV2     T09MRECV QNAME=IPRL,                            X
               OPTIONS=(LL),                                 X
               TRNSTAT=(TERM)
.
.
.
*
*        Send tool definitions required for the IVP to run
*
         T09MSEND QNAME=IPSF,                                X
               PORT=1350,                                    X
               IPNAME=127.0.0.1,                             X
               OPTIONS=(RAW,FILE),                           X
               TRNSTAT=TERM,                                 X
               APISTAT=(CONN,TERM)
*
         T09MSEND QNAME=IPSL,                                X
               PORT=1351,                                    X
               IPNAME=127.0.0.1,                             X
               OPTIONS=(LL),                                 X
               TRNSTAT=TERM,                                 X
               APISTAT=(CONN,TERM)
.
.
.
END
```

Sample Configurations with Matching Receive and Send Tools Using the MROCPT Feature

This is an example of how the Unicenter SOLVE:CPT listen, receive and send tools can be tied together in a CPTMRO environment. Below are some excerpts from the configuration table sample T09CONMR and the T09MRO00 configuration EXEC for CPTMRO, these members are distributed in T09SAMP library. These particular samples show the setup for the Unicenter SOLVE:CPT MRO feature that is used for the UNICENTER SOLVE:CPT IVP for MRO. If you are interested in configuring this same environment without the CPTMRO feature, see the preceding diagram.

Notice the correlation between the two IVPRECV1s and the two IVPRECV2s.

- The lines and arrows on the left side of the diagram show the correlation between the listen tool that is defined in the CPTMRO configuration using the RECEIVEPARM *label* parameter on the DEFINE SESSION object, to the assembler language label that is on the receive tool (TO9MRECV) definition in the CICS centric CPT configuration table. This is the required parameter method that Unicenter SOLVE:CPT associates a listen tool and a receive tool together.
- The lines and arrows on the right side of the diagram show the correlation between the send tool (TO9MSEND) remote PORT= parameter in the CICS centric UNICENTER SOLVE:CPT configuration table, and how it ties to the local PORT parameter on the DEFINE LISTENER object in the CPTMRO configuration. This is an example of how the send tool is used with the UNICENTER SOLVE:CPT IVP. However, in a production environment, this remote server port on the send tool is normally remote from the MVS CICS environment where you are configuring the send tool.

  Note: This is different from the receive tool in that this correlation is not required and is only being used for an example.

T09MRO00 Sample Member Excerpts from T09SAMP Library

```
DEFINE LISTENER     LISTEN01    PORT          1450
                                TCPIPJOB      TCPIP
                                SELECTMETHOD  READY

DEFINE LISTENER     LISTEN02    PORT          1451
                                TCPIPJOB      TCPIP
                                SELECTMETHOD  CIRCULAR

DEFINE SESSION      SESS01      LISTENER      LISTEN01
                                CONNECTION    CONNECT01
                                TRANSID       IPTR
                                RECEIVEPARM   IVPRECV1

DEFINE SESSION      SESS02      LISTENER      LISTEN02
                                CONNECTION    CONNECT01
                                TRANSID       IPTR
                                RECEIVEPARM   IVPRECV2
```

T09CONMR Sample Member Excerpts from T09SAMP Library

```
         T09MCICS TRANSID=(IPPR,IPTL,IPST,IPT2),
               TRCSSN=ACTR,
               JOBNAME=TCPIP

IVPRECV1 T09MRECV QNAME=IPRF,
               OPTIONS=(RAW,FILE),

IVPRECV2 T09MRECV QNAME=IPRL,
               OPTIONS=(LL),

         T09MSEND QNAME=IPSF,
               PORT=1450,
               IPNAME=127.0.0.1,
               OPTIONS=(RAW,FILE),

         T09MSEND QNAME=IPSL,
               PORT=1451,
               IPNAME=127.0.0.1,
               OPTIONS=(LL),
```

T09MSEND Macro

The T09MSEND macro defines the queues that are processed by the CICS/Tools. User programs place data in the CICS transient data queues referenced in the T09MSEND macros. Data is processed from the queues according to the TRIGLEV= and TRANSID= parameters in the DCT.

Define the queues with TRIGLEV parameters so CICS will run the transactions. The T09TSEND transaction (IPTS) scans T09CONFG for a matching entry for the specified transient data queue. Data is read from the queue and a connection is built to the specified IP host and port number. The data from the queue is sent over this connection as specified in the OPTIONS keyword.

See Sample Configuration Table with Matching Receive and Send Tools for a diagram of how the Send and Receive tools tie together for the CPT IVP. Also, see the default IVP configuration sample, T09CONCP, in the T09SAMP library for an example of how the two tools tie together using the PORT parameter.

T09MSEND Syntax

```
T09MSEND   QNAME = name
   [, APISTAT = ( [ CONN ] [, TERM ] ) ]
   [, DNR = YES | NO ]
   [, IPNAME = a.b.c.d ]
   [, OPTIONS = ( [ IPNAME ] [, LL | SEP | ALL | FILE ] [, TRANS | RAW ] ) ]
   [, PORT = ( number ) ]
   [, RCVBUF = ( number, size ) ]
   [, SEP = ( CRLF | CR | LF | X'xx..' ) ]
   [, SERVICE = ( name )
   [, SNDBUF = ( number, size ) ]
   [, STIMEOUT = P | E ]
   [, TRANSTBL = tblname ]
   [, TRNSTAT = ( [ TERM ] ) ]
```

where:

APISTAT

Defines statistic logging options for the API level code of the Unicenter SOLVE:CPT routines.

CONN	Specifies that a message be generated upon establishment of either a server or client connection. This message contains protocol address information.
TERM	Specifies that a statistic message be generated upon termination of a connection. This message contains TRUE data transfer information.

APITRAC

TCPEEP tracing makes the APITRAC parameter obsolete. It is left in for backward compatibility. However, it does not effect application execution.

DNR=Y|N	Indicates whether to resolve remote IP addresses into IP names with DNR calls. Default: NO.	
IPNAME	*a.b.c.d* Defines the host name or an internet address of the server. This parameter can be specified as an internet domain name, as in NIC.DDN.MIL, or as an internet address in dotted decimal form, such as 192.112.36.5.	
OPTIONS	Defines how data from a transient data queue will be processed before it is sent on a TCP/IP connection.	
	IPNAME	Specifies the first record in the transient data queue is to be interpreted as the IPNAME for the connection. Optionally, this record can contain a port number. ***Important!*** *This record is not sent on the TCP connection.* **Note:** The first few bytes of the data, up to a comma, space, or CRLF, are interpreted as a domain name to which to send the data. If a comma appears after the domain name, the next few bytes, up to a space or CRLF, must be numeric, and are interpreted as a port number to which to send the data and the PORT= *keyword* is ignored. For example the string "mvshost.ca.com,1234" establishes the port number as 1234 and overrides the specified value on the PORT parameter.
	LL	Specifies that a two-byte binary length field should be placed before the data read from the CICS transient data queue and sent. The length does not include the LL field.
	SEP	Specifies that a separator sequence will be placed after a record is read from the CICS transient data queue.
	ALL	Specifies that the entire contents of the CICS transient data queue should be read and sent on a single TCP session with no indication of record boundaries.
	FILE	Specifies that the data read from the CICS transient data queue be sent as one TCP session and then closed. The next READQ is sent on a separate TCP session.
	TRANS or RAW	Specifies whether (TRANS) or not (RAW) to translate the data from EBCDIC to ASCII before transmitting to remote system.

PORT	Required. Defines the remote host's server well-known port number. The *number* defines the server transport provider port number. In the special and rare case when OPTIONS=IPNAME is used with a port number being passed along with the DNS name this PORT parameter is ignored. The port is overridden with the port number that was passed as part of the DNS string. For example, the string "mvshost.ca.com,1234" establishes the port number as 1234 and overrides the specified value on the PORT parameter. See the OPTIONS=IPNAME parameter above for greater detail on how to use this feature of the send tool.
QNAME	Required. Defines the transient data queue name that contains data to be sent on a TCP/IP connection. The *name* specifies the queue name from which to receive data.
RCVBUF	Defines buffering space for the Unicenter SOLVE:CPT API routines and space is used to retrieve data from TCP/IP. The buffer number and size are multiplied to determine total receive storage allocation. **Note:** The SEND tool does not use this field.

number	Only specify one. Adding extra buffers wastes storage and does not improve performance. Default: One.
size	Specifies the maximum size of an input or receive buffer. The combination of these must not exceed 61 KB. Default: 1024.

SEP	Keyword that only applies if OPTIONS=SEP. It defines the characters that are appended to the end of a transient data queue record.	
	CRLF	Specifies that records are separated with the ASCII carriage return-line feed character. The ASCII character for CRLF is x‘0D0A’.
	CR	Specifies that records be separated with the ASCII carriage return character. The ASCII character for CR is x‘0D’.
	LF	Specifies that records be separated with the ASCII line feed character. The ASCII character for LF is x‘0A’.
	X‘xx..’	Specifies that records be separated with a specified hexadecimal string. The string must be two or four hexadecimal digits long. Data is sent from the TCP connection and a separator sequence is placed on the end of the record. Maximum length of one record: 32767. The terminator sequence is not translated.
SERVICE	This field is obsolete. It is may be specified but it is not used. **Note:** Sites must use the PORT parameter.	
SNDBUF	Defines buffering space for the Unicenter SOLVE:CPT routines. This buffer space is used to transmit data to TCP/IP. The buffer number and size are multiplied to determine total send storage allocation. The combination of these must not exceed 61 KB.	
	number	Only specify one. Adding extra buffers wastes storage and does not improve performance. Default: One.
	size	Specifies the maximum size of an output or send buffer.
	Default: 1024.	

STIMEOUT	This parameter states whether these sessions will participate in either the GIVE or session inactivity timeout STEAR process. Setting STIMEOUT=P means these sessions will participate in either the GIVE or session inactivity timeout STEAR process. Setting STIMEOUT=E means these sessions will be excluded from the STEAR process. There will not be any GIVE or session inactivity timeout for these sessions. Default: P (Participate in GIVE and session inactivity timeout).	
TRANSTBL	Defines an alternate translation table to use when processing data on connections to this port, otherwise, the default translation table specified on the T09MCICS statement is used.	
	tblname	Specifies the name of the ASCII to EBCDIC translation table for this entry. It overrides the default specified in the T09MCICS statement. For more information about customizing the table, see the "Translation Tables" chapter.
TRNSTAT	Defines statistic logging options for transaction level (automated tool) code of the Unicenter SOLVE:CPT routines.	
	TERM	Specifies that a statistic message be generated upon termination of a connection. This message contains transaction data transfer information.
TRNTRAC	TCPEEP tracing has made the TRNTRAC parameter obsolete. It is left in for backward compatibility. However, it will not effect application execution.	

T09MSLCT Macro

Specifies that the SELECT tool transaction be started at Unicenter SOLVE:CPT initialization.

Note: This macro needs to be specified only if the SELECT tool will be used by any receive program.

T09MSLCT Syntax

```
T09MSLCT TRANSID = name
    , MAXCONN = number
```

where:

TRANSID	Specifies the name of the SELECT tool transaction. This is the transaction that executes program T09TSLCT. Default as distributed: IPSL.
MAXCONN	Specifies the maximum number of concurrent receive transactions using the SELECT tool. The SELECT tool program uses this value to allocate storage (maxconn * 8) to keep track of outstanding receive transactions. **Note:** Since the SELECT tool transaction owns the tokens (endpoints) for any user-application transaction having done a GIVE API call with the AFMOPSEL option, if the SELECT tool transaction is cancelled, then all the tokens (endpoints) are closed, and the connections terminated.

T09MCMDS Macro

The T09MCMDS macro defines the interface from Unicenter NetMaster to Unicenter SOLVE:CPT.

Note: Coding this macro to enable the interface generates one long running transaction in CICS/TS.

IBM TCP/IP only: You need to define port security in the PORT section of the profile.tcpip data set as follows:

```
1234 TCP cicsprod
```

Where:

1234	PORT=1234 parameter.
cicsprod	Name of your CICS/TS started task.

Unicenter TCPaccess running port security: For T09MCMDS coded in the T09CON*xx* configuration file, you may need to define port security in the PORT section of the TCPBND*xx* configuration file. Sample PORT statement to reserve TCP port 1234 with associated job CICSPROD in the TCPBND*xx* configuration file:

```
PORT NUM(1234)
     PROTO(TCP)
     JOBN(CICSPROD)
     ACCESS(SHR)
```

Where:

1234	PORT=1234 parameter.
cicsprod	Name of your CICS/TS started task.

T09MCMDS Syntax

```
T09MCMDS PORT=port,                 X ]
   [, TRANSID=transid,              X ]
   [, SECURITY=secopt,              X ]
   [, SECNAME=secentity,            X ]
   [, LOG=logopt,                   X ]
   [, MSOCK=maxsock,                X ]
   [, QLSTN=qlstn,                  X ]
   [, TERMID=termid                   ]
```

LOG=*logopt*

Specifies whether non-critical command server events are recorded in the log.

Specify one or more of the following keywords within parentheses separated by commas:

SEC	Record security violation messages in the log.
INFO	Record informational messages in the log.
ERROR	Record non-critical error messages in the log.

Note: Critical error messages are always recorded in the log regardless of *logopt*. To record **all** events, specify LOG=(SEC,INFO,ERROR).

Default: ERROR.

MSOCK=*maxsock*

Specifies the maximum number of concurrent administrative connections supported by the command server.

Valid values: A number between 50 and 2000.

Default: 50.

PORT=*port*

Required. The port on which the Unicenter NetMaster command processor listens.

Default: None.

QLSTN=qlstn

Specifies the listen queue depth (backlog).

This is the maximum number of concurrent connection requests permitted in the listen request queue. When this queue is full, subsequent connection requests are discarded.

Default: Five.

SECNAME=*secentity*	Name of the external security system (ESM) general resource entity the command processor uses to verify user command authority. The command processor issues an ESM/SAF call to verify users have READ or UPDATE access against this resource.

The name of this resource can be up to 44 characters and must be filed in the ESM's general resource profile (for example, FACILITY).

- If users have READ access, they are allowed to display session, server, and global statistics
- If users have update access, in addition to displaying session, server, and global statistics, they are also allowed to start and stop transactions, sessions, servers and applications

Note: If omitted, the default, $SKTVIEW.CICS.CMDAUTH, is used.

SECURITY=Y|N

SECURITY=Y	Specifies whether the command server validates a user ID and password combination for all users. Should the user provide a valid user ID/password combination then the user's authority to the SECNAME entity is validated. The terminal configured in the TERMID parameter must be available for use in the CICS/TS region whenever SECURITY=Y is set.
SECURITY=N	Allows any user of the command server to use all commands.

Important! *You may need to disable security in a testing environment, but it is strongly recommended that you always enable security in a production environment.*

Default: Y.

TERMID=*termid*	Specifies the name of the terminal with which to associate the command server. See the Terminal Control (TCT) Entries in the "CICS/TS Resource Definition Reference" chapter for more information. Default: TCMD.
TRANSID=*transid*	The Unicenter NetMaster command processor transaction ID. This parameter must match the definition for program T09TCMDS in the CICS/TS RDO definitions. This is the CICS/TS transaction used during Unicenter SOLVE:CPT initialization to launch the command server. Default: IPCP.

T09MTRAN Macro

The T09MTRAN macro defines a CICS transaction that can be started by Unicenter SOLVE:CPT. It is an excellent mechanism to start non-T09MLSTN servers after Unicenter SOLVE: CPT is properly initialized.

Note: See the appendix “T09MTRAN Programming Notes” in the *Programmer's Reference* for information describing T09MTRAN macro and its parameters.

T09MTRAN Syntax

```
T09MTRAN
   [  TRANSID = transaction to start ]
   [, USERID = userid ]
   [, TERMID = CICS terminal ]
   [, IMMED = YES | NO ]
   [, ID = unique 1-8 character ID ]
   [, APPLID = CICS VTAM APPLID ]
   [, PORT = ( number ) ]
   [, BACKLOG = num_backlog_queue ]
   [, ACCTIME = accept() timeout ]
   [, REATIME = read() timeout ]
   [, GIVTIME = givesocket() timeout ]
   [, NUMSOCK = num_sockets ]
   [, MINMSGL = miminium message length ]
   [, TRANTRN = YES | NO ]
   [, TRANUSR = YES | NO ]
   [, SCTYEXIT = exit-program-name ]
   [, WLMGN1 = WLM Group name 1 ]
   [, WLMGN2 = WLM Group name 2 ]
   [, WLMGN3 = WLM Group name 3 ]
```

Note: There is limit of 255 T09MTRAN entries in the T09CON*xx* configuration file.

ACCTIME

Accept timeout in seconds, found in the CFLLTME field addressed by the CFG0000 DSECT from the LSTCFGDA pointer.

Note: Specifying the ACCTIME parameter forces the LSTP parameter DSECT to be passed to the TRANSID transaction.

Valid values: 0-999.

Default: 60.

APPLID

Specifies the name of the CICS VTAM APPLID to run the transaction at product startup. If you let the APPLID default, it runs at product startup (depending on the IMMED specification). If you place a parameter on the APPLID field, then the APPLID must match the CICS VTAM APPLID in the running CICS region before the transaction can be started in the region at product startup.

Default: Eight blanks.

BACKLOG	Listen Backlog Queue to be found in the CFLBKLOG field addressed by the CFG0000 DSECT from the LSTCFGDA pointer. **Note:** Specifying the BACKLOG parameter forces the LSTP parameter DSECT to be passed to the TRANSID transaction. Valid values: 0-999. Default: 20.
ID	Unique one- to eight-character ID used to uniquely identify an entry. If you allow T09MTRAN to default, it generates unique character IDs of the form ID appended with the instance of the T09MTRAN macro in the T09CON*xx* configuration file. Default: ID appended with the instance of the T09MTRAN macro in the T09CON*xx* configuration file.
IMMED= YES|NO	States whether the transaction should be started at product startup. Default: YES.
GIVTIME	givesocket() timeout in seconds, found in the CFLGTME field addressed by the CFG0000 DSECT from the LSTCFGDA pointer. **Note:** Specifying the GIVTIME parameter forces the LSTP parameter DSECT to be passed to the TRANSID transaction. Valid values: 0-999. Default: 60.
MINMSGL	Specifies the minimum input message length, found in the CFLNMIN field addressed by the CFG0000 DSECT from the LSTCFGDA pointer. **Note:** Specifying the MINMSGL parameter forces the LSTP parameter DSECT to be passed to the TRANSID transaction. Valid values: 4-99. Default: Four.

NUMSOCK	Specifies the maximum number of concurrent connections supported, found in the CFLNSOCK field addressed by the CFG0000 DSECT from the LSTCFGDA pointer. **Note:** Specifying the NUMSOCK parameter forces the LSTP parameter DSECT to be passed to the TRANSID transaction. Valid values: 50-2000. Default: 50.
PORT	Port Number found in the CFLPORT field addressed by the CFG0000 DSECT from the LSTCFGDA pointer. **Note:** Specifying the PORT parameter forces the LSTP parameter DSECT to be passed to the TRANSID transaction. Valid values: 1-65535. Default: None.
REATIME	Read timeout in seconds, found in the CFLRTME field addressed by the CFG0000 DSECT from the LSTCFGDA pointer. **Note:** Specifying the ACCTIME parameter forces the LSTP parameter DSECT to be passed to the TRANSID transaction. Valid values: 0-32767. Default: Zero.
SECEXIT	Security exit program name, found in the CFLSECEX field addressed by the CFG0000 DSECT from the LSTCFGDA pointer. **Note:** Specifying the MINMSGL parameter forces the LSTP parameter DSECT to be passed to the TRANSID transaction. Default: Eight blanks.
TERMID	Specifies the name of the terminal to associate with the started transaction. The TERMID parameter cannot be specified with the USERID parameter. Default: None.
TRANSID	Required. Transaction ID to start. Default: None.

TRANTRN= YES|NO	States whether the transaction ID should be translated, found in the CFLOPTTR field addressed by the CFG0000 DSECT from the LSTCFGDA pointer. **Note:** Specifying the TRANTRN parameter forces the LSTP parameter DSECT to be passed to the TRANSID transaction Default: None when no other CFG0000 parameters are specified. The default is YES when other CFG0000 parameters are specified.
TRANUSR= YES|NO	States whether user data should be translated, found in the CFLOPTUD field addressed by the CFG0000 DSECT from the LSTCFGDA pointer. **Note:** Specifying the TRANUSR parameter forces the LSTP parameter DSECT to be passed to the TRANSID transaction Default: None when no other CFG0000 parameters are specified. YES when other CFG0000 parameters are specified.
USERID	User ID to use when starting the transaction. This allows the transaction to inherit the security permissions of the specified user ID. **Note:** The USERID parameter cannot be specified with the TERMID parameter. Default: None.
WLMGN1	WLM Group name 1, found in the CFLWLMN1 field addressed by the CFG0000 DSECT from the LSTCFGDA pointer. **Note:** Specifying the WLMGN1 parameter forces the LSTP parameter DSECT to be passed to the TRANSID transaction. Default: 12 blanks.
WLMGN2	WLM Group name 2, found in the CFLWLMN2 field addressed by the CFG0000 DSECT from the LSTCFGDA pointer. **Note:** Specifying the WLMGN2 parameter forces the LSTP parameter DSECT to be passed to the TRANSID transaction. Default: 12 blanks.
WLMGN3	WLM Group name 3, found in the CFLWLMN3 field addressed by the CFG0000 DSECT from the LSTCFGDA pointer. **Note:** Specifying the WLMGN3 parameter forces the LSTP parameter DSECT to be passed to the TRANSID transaction. Default: 12 blanks.

T09MANAG Macro

Defines the session management related parameters.

T09MANAG Syntax

```
T09MANAG

   [  GTIMEOUT = GIVE timeout ]
   [, PURGETO  = N | Y ]
   [, STIMEOUT = Session timeout ]
   [, TTIMEOUT = IPUE ]
```

GTIMEOUT	GTIMEOUT sets the maximum number of seconds a session will allow a CPT session to be in the GIVE state before a corresponding TAKE should take ownership of a session. The CPT STEAR process monitors CPT sessions to see if any session is in the GIVE state and has exceeded the GIVE timeout. A CPT session which exceeds the GIVE timeout parameter is considered to be abandoned. The CPT STEAR process will CLOSE any CPT session which exceeds the GIVE timeout. Specifying GTIMEOUT=0 means that there is no GIVE timeout for CPT sessions. Session will never time out in the GIVE state. Valid values: 0-14439. Default: 30.
PURGETO	Specifies whether the STEAR process should purge any CICS task associated with a CPT session, which has exceeded the session timeout (STIMEOUT) limit. Specifying PURGETO=N means that that the CPT STEAR process will just CLOSE the CPT session and leave the CICS transaction alone. Specifying PURGETO=Y means that that the CPT STEAR process will CLOSE the CPT session and then attempt to purge the CICS transaction, which was associated with the CPT session. Whenever a site sets PURGETO=Y the STEAR TTIMEOUT transaction must have the authority to purge any CICS task which might exceed the session timeout limit. Valid values are: N | Y. Default N.

STIMEOUT	STIMEOUT sets a session inactivity timeout. STIMEOUT sets the maximum number of seconds a session will allow a CPT session to be inactive before issuing another CPT command. The session inactivity timeout starts from the last time a command completes. Sessions that are currently executing a CPT command (for example, RECEIVE) will not be considered as eligible for session inactivity timeout. The CPT STEAR process monitors CPT sessions to see if any session has exceeded the session inactivity timeout. A CPT session which exceeds the session inactivity timeout parameter is considered to be abandoned. The CPT STEAR process will CLOSE any CPT session which exceeds the session inactivity timeout. Specifying STIMEOUT=0 means that there is no session inactivity timeout for CPT sessions. Session will never time out in an inactive state. Server sessions do not participate in the STEAR session inactivity timeout process. Only daughter server sessions or sessions created by CONNECT will be eligible for STEAR management. One does not want a server to be shut down during the off hours due to inactivity. Valid values: 0-14439. Default: 60.
TTIMEOUT	TTIMEOUT names the transaction which is responsible for implementing the STEAR process. STEAR program T09TEPCK needs to be associated with the TTIIMOUT transaction. Default: IPUE

T09MEND Macro

The T09MEND macro terminates lists of configuration parameters.

Important! *This macro has no parameters, but is required and must be the last macro statement in the configuration.*

Chapter 3 CICS/TS Resource Definition Reference

This chapter provides information about the installation of the required CICS/TS Resource Definitions (RDO) needed for Unicenter SOLVE:CPT.

This chapter is meant to be a reference for the CICS definitions, not an installation guide. The steps required to install these definitions are documented in the *Getting Started*.

It includes these topics:

Note: See *Getting Started* for software installation instructions.

Unicenter SOLVE:CPT definition statements must be added to the CICS/TS Resource Definition Online (RDO) and to resource definition tables. Program, transaction ID, and destination definition statements are required. Some destination and PLT definition statements are recommended, but are optional.

You must define the Unicenter SOLVE:CPT program names for system initialization, termination, Task-Related User Exits (TRUEs), and configuration to CICS/TS RDO. These programs require transaction IDs that must be defined. The transaction IDs are distributed with default values, but can be modified during Unicenter SOLVE:CPT installation or customization.

A sample DFHCSDUP input file, member name T09RDO, is provided in the T09SAMP library for defining program, transaction, and destination entries using CICS/TS RDO.

Computer Associates recommends that you define the initialization program in the PLTPI table. Alternatively, you can start the initialization program interactively by using its transaction ID (see the chapter "Operations"). The termination program *must* be defined in the PLTSD table. The termination program provides proper release of transport provider connections and Unicenter SOLVE:CPT resources during CICS/TS shutdown. The Unicenter SOLVE:CPT termination and initialization program transaction IDs can be used interactively at a CICS/TS terminal to control Unicenter SOLVE:CPT.

Unicenter SOLVE:CPT uses transient data queues (TDQ) for logging support and for the Unicenter SOLVE:CPT tools facility. Unicenter SOLVE:CPT logs informational errors, statistics, and traces messages to transient data queues. These TDQs can be existing or new queues. If the TDQs are new, they must be defined as shown in sample T09RDO entries.

Unicenter SOLVE:CPT definition statements must be in the appropriate tables either by configuring them, or by making them dynamically available through RDO commands. You may need to restart CICS depending on the installation method you used.

Configuration Table Suffixing

The default configuration table's name is T09CONFG. We recommend that sites configure, assemble and link a T09CONFG member into their T09LOAD or table library. Just copy the sample member T09CONCP from T09SAMP library and rename it to T09CONFG. When a site uses the default T09CONFG configuration table name, it makes product startup simpler.

You can:

- Start the product with the IPST transaction (associated with program T09TSTRT) without any parameters
- Place program T09TSTRT in the PLTPI table without having to code a SIT INITPARM override for the proper configuration parameter override

Multiple configuration tables can reside in a load library. You can link edit the tables with different characters in either the last single position or the last two positions of the configuration table name. This allows multiple CICS/TS regions on one or many hosts to have unique LISTEN tools or NetMaster interfaces defined. Each single-position suffix or two-character position suffix can be specified to the startup transaction IPST when starting the product from a terminal. The suffix can also be passed to the startup program T09TSTRT through a CICS/TS SIT table's INITPARM parameter at CICS/TS initialization time.

Note: A suffix specified dynamically from a CICS/TS terminal takes priority over any INITPARM option.

The suffix is entered as a parameter to the startup transaction:

```
IPST X1
```

Loads configuration table T09CONX1.

Secondary to a CICS/TS terminal start up of Unicenter SOLVE:CPT is starting using a PLTPI entry for the T09TSTRT program. See Program List Table (PLT) Entries for more details on the PLT entries. The default product configuration table is T09CONFG. A site that runs the product with a non-default T09CON*xx* configuration file can pass the two-digit suffix in the CICS/TS SIT table INITPARM parameter overrides.

For example, place the following INITPARM entry in the SIT overrides, shows you how to start the product using the T09CONA2 configuration table during CICS startup:

```
INITPARM=(T09TSTRT='A2')
```

Loads configuration table T09CONA2.

If you use a configuration table other than T09CONFG, it must be defined to CICS/TS RDO definitions as a program as the following example shows.

```
CEDA DEF PROG(T09CONA2) LANG(ASSEMBLER) G(T09CPT)
```

If neither override method is used, the default table name, T09CONFG, is loaded.

System Initialization Tasks (SIT) Entries

During Unicenter SOLVE:CPT installation or customization, pay close attention to the maximum active tasks (AMXT) and maximum number of concurrent tasks (MXT) values. The Unicenter SOLVE:CPT server application, the Listen Tool, is designed to be a long-running CICS transaction. Additionally, a server can be written to start any number of concurrent data processing tasks, the automated listening transaction is such a task. Therefore, installations should monitor active Unicenter SOLVE:CPT applications for the number of concurrent, active tasks.

SIT Override for Starting Unicenter SOLVE:CPT from the PLTPI Table

The default product configuration table is T09CONFG. A site that runs the product with a non-default T09CON*xx* configuration file may pass the two-digit suffix in the CICS/TS SIT table INITPARM parameter overrides. For example, place the following INITPARM entry in the SIT overrides to start the product using the T09CONA2 configuration table during CICS startup:

```
INITPARM=(T09TSTRT='A2')
```

Loads configuration table T09CONA2 at product startup.

SIT Overrides Required for CPTMRO

Both ISC=YES and IRCSTRT=YES must be set in your CICS/TS SIT startup parameters when a site will be running the CPTMRO Server address space outside of CICS. CPTMRO uses the EXCI facility to pass sessions into the CICS address space.

Program List Table (PLT) Entries

The product can use the standard CICS facilities to be started and terminated with CICS.

Starting Unicenter SOLVE:CPT During CICS Startup

The initialization routine resource definition shown below can be included in the PLTPI. Depending on the release, the Unicenter SOLVE:CPT initialization routine must appear after the entry statement for DFHDELIM or during the third stage of initialization. The resource definition statement for the T09TSTRT program is optional, but is generally recommended.

A default PLTPI table entry can be copied from the T09SAMP library member T09PLTPI.

```
*-------------------------------------------------------------------*
*                                                                   *
*        Socket Management & CPT  PLT (PLTPI=) ENTRY                *
*                                                                   *
*        THESE PLT ENTRIES ARE FOR THE SYSTEM INITIALIZATION PLT    *
*        TABLE.                                                 *
*                                                                   *
*-------------------------------------------------------------------*
*                                                                   *
*        STARTUP ENTRY                                              *
*                                                                   *
         DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
*
         DFHPLT TYPE=ENTRY,        Socket Management/CPT START UP
               PROGRAM=T09TSTRT
*
```

Note:

- Unless a site uses an INITPARM SIT override (INITPARM=(T09TSTRT='CP') for T09CONCP) the product tries to use the T09CONFG configuration file during product startup out of the PLTPI member

- See Configuration Table Suffixing earlier in this chapter for details on alternate configuration table names

Stopping Unicenter SOLVE:CPT During CICS Termination

This termination routine resource definition must be included in the shutdown section of the PLTSD. Depending on the release of CICS/TS, the Unicenter SOLVE:CPT termination routine must appear **before** the entry statement for DFHDELIM or during the first stage of termination. This program provides proper termination of the Unicenter SOLVE:CPT Interface (the Interface) during CICS/TS shutdown. This includes the release of all transport provider connections and disabling the Interface.

A default PLTSD table entry can be copied from the T09SAMP library member T09PLTSD.

```
*---------------------------------------------------------------------*
*                                                                     *
*        Socket Management / CPT PLT (PLTSD=) ENTRY                   *
*                                                                     *
*        THIS PLT ENTRY IS FOR THE SYSTEM TERMINATION PLT TABLE.  *
*                                                                     *
*---------------------------------------------------------------------*
*
         DFHPLT TYPE=ENTRY,                                           X
               PROGRAM=T09TTERM
*
         DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
```

Sample CICS/TS RDO Member T09RDO

The default CICS/TS RDO entries can be copied from the T09SAMP library member T09RDO, which is shown below. Installing these definitions is required for the product. As noted in the comments at the top of the T09RDO member, a **?** was inserted in front of the delete group command to prevent accidental deletion. It is recommended that Unicenter SOLVE:CPT for CICS have its own RDO group, so that CICS/TS RDO entries can be easily deleted and refreshed with any new releases of the product.

Note:

- If you are upgrading from an older version of Unicenter SOLVE:CPT you must still run this step, since changes have been made to the RDO definitions. If you do not update your definitions, you chance exposures to S0C1s and S0C4s at initialization.
- If you follow this procedure, blank out the **?**, and install the group below. This group must be part of the RDO list that starts up this CICS/TS region.

```
*-----------------------------------------------------------------------*
*                                                                       *
*   Socket Management and CICS PROGRAMMER'S TOOLKIT RDO ENTRIES        *
*                                                                       *
*-----------------------------------------------------------------------*
*                                                                       *
* The below "?" is intended to cause a failure so that this sample     *
* does not accidentally delete an existing RDO group.  If you desire   *
* to start with fresh definations which are needed with this new       *
* version you will need to blank out the "?".  Make sure first,        *
* that any extra defintions within this group are saved within         *
* another group if they are needed.  If you simply want to add the     *
* new enties to the existing group then delete the following line and  *
* submit this member as is, this will cause existing enties to fail    *
* for dups and the new enties to be added.                             *
*                                                                       *
? DELETE GROUP (T09CPT)
*                                                                       *
*-----------------------------------------------------------------------*
*                                                                       *
*   Socket Management and CICS PROGRAMMER'S TOOLKIT PROGRAMS           *
*                                                                       *
*-----------------------------------------------------------------------*
*
 DEFINE PROG (T09COMON) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09CONFG) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09CONCP) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09CONEZ) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TSTRT) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
        EXECK (CICS)
 DEFINE PROG (T09TTERM) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
        EXECK (CICS)
 DEFINE PROG (T09TCMDS) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
        EXECK (CICS)
 DEFINE PROG (T09TLOID) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
        EXECK (CICS)
 DEFINE PROG (T09TLCAF) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TLSTN) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TLST2) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TMROS) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TQUSV) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TRECV) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TSEND) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TSLCT) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TULST) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09XENG)  LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TIPCK) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TIPEC) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TIPES) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09PACSS) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09IUEXT) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09ATADD) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09ABDTL) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09ACNFG) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09ADMGR) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09AEXIT) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09AGENT) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09AOLWT) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09AHELP) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09ATLST) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09AMAIN) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09AMENU) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09AOCTL) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09APING) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09AQCLS) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09ARSTQ) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09ATSND) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09ATROP) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09AUTIL) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09AWTCH) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09AYANK) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09AZAPS) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09FCLOS) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09FCONN) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09FGIVE) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09FLFTP) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09FLSTN) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09FRCFR) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09FRECV) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09FSEND) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09FSLCT) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09FSNTO) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09FTAKE) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09FXLAT) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09MAPS)  LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09MAPT)  LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09MAPU)  LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09NMEVX) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TASC1) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TASC2) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09TCFCM) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
        EXECK (CICS)
 DEFINE PROG (T09TCFDG) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
        EXECK (CICS)
 DEFINE PROG (T09TCFDM) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
        EXECK (CICS)
 DEFINE PROG (T09TCFRM) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
        EXECK (CICS)
 DEFINE PROG (T09TEPCK) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
        EXECK (CICS)
 DEFINE PROG (T09ETRUE) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (EZACIC01) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
 DEFINE PROG (T09CIC01) LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
*
*--------------------------------------------------------------------*
*                                                                    *
*   Socket Management and CICS PROGRAMMER'S TOOLKIT TRANSACTIONS     *
*                                                                    *
*--------------------------------------------------------------------*
*
 DEFINE TRANS (IPLF) PROG (T09TLCAF) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPST) PROG (T09TSTRT) TASKDATALOC (ANY) GROUP (T09CPT)
        TASKDATAK (CICS)
 DEFINE TRANS (IPPR) PROG (T09TTERM) TASKDATALOC (ANY) GROUP (T09CPT)
        TASKDATAK (CICS)
 DEFINE TRANS (IPQU) PROG (T09TQUSV) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPTL) PROG (T09TLSTN) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPT2) PROG (T09TLST2) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPTR) PROG (T09TRECV) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPTS) PROG (T09TSEND) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPSL) PROG (T09TSLCT) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPUE) PROG (T09TEPCK) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPUL) PROG (T09TULST) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPCK) PROG (T09TIPCK) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPCS) PROG (T09PACSS) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAA) PROG (T09ATADD) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAB) PROG (T09ABDTL) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAC) PROG (T09ACNFG) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAE) PROG (T09AQCLS) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAG) PROG (T09AGENT) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAI) PROG (T09AOLWT) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAK) PROG (T09AHELP) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAL) PROG (T09ATLST) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAM) PROG (T09AMAIN) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAN) PROG (T09AMENU) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAO) PROG (T09AOCTL) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAP) PROG (T09APING) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAQ) PROG (T09AQCLS) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAR) PROG (T09ARSTQ) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAS) PROG (T09ATSND) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAT) PROG (T09ATROP) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAU) PROG (T09AUTIL) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAW) PROG (T09AWTCH) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAY) PROG (T09AYANK) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPAZ) PROG (T09AZAPS) TASKDATALOC (ANY) GROUP (T09CPT)
 DEFINE TRANS (IPEC) PROG (T09TIPEC) TASKDATALOC (ANY) GROUP (T09CPT)
        DESC(EZASOKET IVP CLIENT TASK CONNECT W IPES)
 DEFINE TRANS (IPES) PROG (T09TIPES) TASKDATALOC (ANY) GROUP (T09CPT)
        DESC(EZASOKET IVP SERVER TASK)
 DEFINE TRANS (IPFG) PROG (T09TCFDG) TASKDATALOC (ANY) GROUP (T09CPT)
        TASKDATAK (CICS)
 DEFINE TRANS (IPCP) PROG (T09TCMDS) TASKDATALOC (ANY) GROUP (T09CPT)
        TASKDATAK (CICS)
 DEFINE TRANS (IPFL) PROG (T09TLOID) TASKDATALOC (ANY) GROUP (T09CPT)
        TASKDATAK (CICS)
 DEFINE TRANS (IPFD) PROG (T09TCFDM) TASKDATALOC (ANY) GROUP (T09CPT)
        TASKDATAK (CICS)
 DEFINE TRANS (IPFR) PROG (T09TCFRM) TASKDATALOC (ANY) GROUP (T09CPT)
        TASKDATAK (CICS)
*                                                                      *
*   THESE DCT-TYPE ENTRIES BELOW ARE FOR THE ERROR, STATISTIC AND      *
*   TRACE FACILITY RESPECTIVELY.  THE INDIRECT OPTION IS SELECTED      *
*   USING THE 'CSML' TRANSIENT DATA QUEUE.  AN INSTALLATION SHOULD     *
*   VERIFY THE 'CSML' DCT ENTRY OR SELECT AN ALTERNATIVE.              *
*   THESE ENTRIES ARE REQUIRED.                                        *
*                                                                      *
 DEFINE TDQUEUE(ACER) TYPE(INDIRECT) INDIRECTNAME(CSML) GROUP(T09CPT)
 DEFINE TDQUEUE(ACST) TYPE(INDIRECT) INDIRECTNAME(CSML) GROUP(T09CPT)
 DEFINE TDQUEUE(ACTR) TYPE(INDIRECT) INDIRECTNAME(CSML) GROUP(T09CPT)
*                                                                      *
*   THESE DCT-TYPE ENTRIES ARE NEEDED FOR THE CPT SAMPLE                *
*   CONFIGURATION T09CONCP TO WORK.  THESE ENTRIES SUPPORT             *
*   THE INSTALLATION VERIFICATION PROCEDURE (IVP) AND                  *
*   THE OTHER SEND/RECIEVE TOOL SAMPLES                                *
*                                                                      *
*   ********>>>>>>>> THESE ENTRIES ARE OPTIONAL. <<<<<<<<********      *
*                                                                      *
*   RECEIVE (SERVER) OPTION=FILE                                       *
 DEFINE TDQUEUE(IPRF) TYPE(INTRA) TRIGGERLEVEL(0)
        RECOVSTATUS(NO)      ATIFACILITY(FILE)    GROUP(T09CPT)
*   RECEIVE (SERVER) OPTION=LL                                         *
 DEFINE TDQUEUE(IPRL) TYPE(INTRA) TRIGGERLEVEL(0)
        RECOVSTATUS(LOGICAL) ATIFACILITY(FILE)    GROUP(T09CPT)
*   RECEIVE (SERVER) OPTION=ALL                                        *
 DEFINE TDQUEUE(IPRA) TYPE(INTRA) TRIGGERLEVEL(0)
        RECOVSTATUS(LOGICAL) ATIFACILITY(FILE)    GROUP(T09CPT)
*   RECEIVE (SERVER) OPTION=SEP (CR/FL)                                *
 DEFINE TDQUEUE(IPRS) TYPE(INTRA) TRIGGERLEVEL(0)
        RECOVSTATUS(LOGICAL) ATIFACILITY(FILE)    GROUP(T09CPT)
*   SEND (CLIENT) OPTION=FILE                                          *
 DEFINE TDQUEUE(IPSF) TYPE(INTRA) TRIGGERLEVEL(1) TRANSID(IPTS)
        RECOVSTATUS(LOGICAL) ATIFACILITY(FILE)    GROUP(T09CPT)
*   SEND (CLIENT) OPTION=LL                                            *
 DEFINE TDQUEUE(IPSL) TYPE(INTRA) TRIGGERLEVEL(1) TRANSID(IPTS)
        RECOVSTATUS(LOGICAL) ATIFACILITY(FILE)    GROUP(T09CPT)
*   SEND (CLIENT) OPTION=ALL                                           *
 DEFINE TDQUEUE(IPSA) TYPE(INTRA) TRIGGERLEVEL(1) TRANSID(IPTS)
```

```
        RECOVSTATUS(LOGICAL) ATIFACILITY(FILE)    GROUP(T09CPT)
*   SEND (CLIENT) OPTION=SEP (CR/FL)                                 *
 DEFINE TDQUEUE(IPSS) TYPE(INTRA) TRIGGERLEVEL(1) TRANSID(IPTS)
        RECOVSTATUS(LOGICAL) ATIFACILITY(FILE)    GROUP(T09CPT)
*
*-------------------------------------------------------------------*
*                         LIST OBJECTS                              *
*-------------------------------------------------------------------*
*
 LIST GROUP(T09CPT) OBJECTS
```

Transient Data Queue (TDQ) Entries

These are the required TDQ entries from the T09RDO member:

```
DEFINE TDQUEUE(ACER) TYPE(INDIRECT) INDIRECTNAME(CSML) GROUP(T09CPT)
DEFINE TDQUEUE(ACST) TYPE(INDIRECT) INDIRECTNAME(CSML) GROUP(T09CPT)
DEFINE TDQUEUE(ACTR) TYPE(INDIRECT) INDIRECTNAME(CSML) GROUP(T09CPT)
```

The required program entries are installed as previously described in Sample CICS/TS RDO Member T09RDO.

Unicenter SOLVE:CPT uses transient data queues for logging support information. The log transient data queues are defined in the T09MCICS macro instruction by the QNAMES parameter. See the “Configuration Reference” chapter for details about these queues.

The ACST, ACTR and ACER are our default transient queues names. The QNAMES parameter on the T09MCICS macro in the T09CON*xx* configuration file can be used to change the QUEUE names used with the product.

Note:

- The Unicenter SOLVE:CPT queues by default are redirected (indirect) to CSML queue, which is often redirected (indirect) to CSSL, which in turn is frequently redirected to SYSOUT. The redirection is expected and works fine. The above is how the sample definitions are shown.
- However, if a data set is the final destination allocated, then the data set should have variable length records of at least 121 bytes.

Program Entries

The required program entries are installed as previously described in Sample CICS/TS RDO Member T09RDO.

Processing program statements must be defined for all Unicenter SOLVE:CPT base product programs.

These programs are:

- System initialization
- Termination
- TRUE routines
- Configuration
- Translation table
- Support routines

All Unicenter SOLVE:CPT base product programs are written in assembler.

Note: Unicenter SOLVE:CPT base product program names cannot be modified.

Use the RDO sample and the information in the following table of Unicenter SOLVE:CPT base product program definitions to define base product programs. This table is in alphabetical order by program name

Program	Language	Description
EZACIC01	Assembler	CICS TRUE for IBM CICS Sockets Compatibility.
T09ATADD	Assembler	Add/Alter tool.
T09ABDTL	Assembler	Browse detail—clients or servers.
T09ACNFG	Assembler	Configuration summary.
T09ADMGR	Assembler	Screen manager.
T09AEXIT	Assembler	Exit interface.
T09AGENT	Assembler	Generate trace.
T09AOLWT	Assembler	Online queue write.
T09AHELP	Assembler	Help presentation.
T09ATLST	Assembler	Listen tool display.
T09AMAIN	Assembler	Main entry manager.

Program	Language	Description
T09AMENU	Assembler	Main menu display.
T09AOCTL	Assembler	Operations menu.
T09APING	Assembler	Ping remote host.
T09AQCLS	Assembler	Close active listeners, or abort connection.
T09ARSTQ	Assembler	Reset trace/error queues.
T09ATSND	Assembler	Send tool display.
T09ATROP	Assembler	Add/alter trace options
T09AUTIL	Assembler	Utilization summary.
T09AWTCH	Assembler	Online queue display.
T09AYANK	Assembler	Online queue release.
T09AZAPS	Assembler	Reset statistics.
T09CIC01	Assembler	CPT CICS TRUE
T09COMON	Assembler	CICS TRUE routines.
T09CONCP	Assembler	Sample Configuration table for Unicenter SOLVE:CPT.
T09CONEZ	Assembler	Sample Configuration table for Unicenter NetMaster Socket Management for CICS.
T09CONFG	Assembler	Default Configuration table name for Unicenter SOLVE:CPT and Unicenter NetMaster Socket Management for CICS.
T09FCLOS	Assembler	CPT API stub routine for the close call.
T09FCONN	Assembler	CPT API stub routine for the connect call.
T09FGIVE	Assembler	CPT API stub routine for the give call.
T09FLFTP	Assembler	CPT API stub routine for the FTP client call.
T09FLSTN	Assembler	CPT API stub routine for the listen call.
T09FRCFR	Assembler	CPT API stub routine for the receivefrom call.
T09FRECV	Assembler	CPT API stub routine for the receive call.
T09FSEND	Assembler	CPT API stub routine for the send call.
T09FSLCT	Assembler	CPT API stub routine for the select call.
T09FSNTO	Assembler	CPT API stub routine for the sendto call.

Program	Language	Description
T09FTAKE	Assembler	CPT API stub routine for the take call.
T09FXLAT	Assembler	CPT API stub routine for the translate call.
T09MAPS	Assembler	Basic mapping support screens.
T09MAPT	Assembler	Basic mapping support screens.
T09MAPU	Assembler	Basic mapping support screens.
T09NMEVX	Assembler	Unicenter NetMaster event exit.
T09TASC1	Assembler	Security exit for the standard listener.
T09TASC2	Assembler	Security exit for the client data listener.
T09TCFCM	Assembler	CPT FTP Client Control Manager.
T09TCFDG	Assembler	CPT FTP Client Data Manager for receive.
T09TCFDM	Assembler	CPT FTP Client Data Manager.
T09TCFRM	Assembler	CPT FTP Client Response Manager.
T09TCMDS	Assembler	Unicenter NetMaster command processor transaction (must run with EXECKey = CICS).
T09TEPCK	Assembler	STEAR program for GIVE and session inactivity timeouts
T09TIPCK	Assembler	CPT/API and CPT Tools IVP transaction.
T09TIPEC	Assembler	EZASOKET API client IVP transaction.
T09TIPES	Assembler	EZASOKET API server IVP transaction.
T09TLCAF	Assembler	Free LCA area at termination after delay.
T09TLOID	Assembler	Initialization processing transaction (must run with EXECKey = CICS).
T09TLSTN	Assembler	Automated listening transaction.
T09TLST2	Assembler	Automated listening transaction phase two.
T09TMROS	Assembler	MRO program.
T09TQUSV	Assembler	Quiesce all active servers.
T09TRECV	Assembler	Automated receive transaction.
T09TSEND	Assembler	Automated send transaction.
T09TSLCT	Assembler	Select tool transaction.
T09TSTRT	Assembler	Initialization transaction (must run with

Program	Language	Description
		EXECKey = CICS).
T09TTERM	Assembler	Termination transaction.
T09TULST	Assembler	Initializes user listen transactions at startup.
T09XENG	Assembler	Translation table.

Transaction Entries

The required transaction entries are installed as previously described in Sample CICS/TS RDO Member T09RDO.

Program control statements must be defined for Unicenter SOLVE:CPT base product transactions. These transactions consist of system initialization, termination, automated tool routines, and support routines.

The Unicenter SOLVE:CPT base product is distributed with default transaction IDs. These transaction IDs are specified within the Unicenter SOLVE:CPT configuration file T09CONcp. If the default transaction IDs are modified then the change must be reflected in the configuration file.

Use RDO and the information in the following table of Unicenter SOLVE:CPT base product transaction ID definitions to define Unicenter SOLVE:CPT base product transaction IDs.

Note: Only the transactions that are noted as *Can be issued directly by a user* should ever be issued by typing their transaction name directly at a CICS terminal. All the other transactions are issued for you through the Administrative Interface (IPAM). If you issue a transaction directly that should not be issued in that fashion, results may be unpredictable.

This table is in alphabetical order by transaction.

Program	Trans ID	Description
T09ATADD	IPAA	Add/alter tool.
T09ABDTL	IPAB	Browse detail—clients or servers.
T09ACNFG	IPAC	Configuration summary. *Can be issued directly by a user.*
T09AQCLS	IPAE	Close connections.
T09AGENT	IPAG	Generate trace.

Program	Trans ID	Description
T09AOLWT	IPAI	Online queue write.
T09AHELP	IPAK	Help presentation.
T09ATLST	IPAL	Listen tool display.
T09AMAIN	IPAM	Main entry manager. *Can be issued directly by a user.*
T09AMENU	IPAN	Main menu misplay.
T09AOCTL	IPAO	Operations menu. *Can be issued directly by a user.*
T09APING	IPAP	Ping remote host.
T09AQCLS	IPAQ	Close connections.
T09ARSTQ	IPAR	Reset trace/error queues.
T09ATSND	IPAS	Send tool display.
T09ATROP	IPAT	Add/alter trace options.
T09AUTIL	IPAU	Utilization summary. *Can be issued directly by a user.*
T09AWTCH	IPAW	Online queue display.
T09AYANK	IPAY	Online queue release.
T09AZAPS	IPAZ	Reset statistics.
T09TEPCK	IPUE	STEAR program for GIVE and session inactivity timeouts
T09TIPCK	IPCK	CPT/API and CPT Tools IVP transaction. *Can be issued directly by a user.*
T09TIPCP	IPCP	NetMaster command processor transaction.
T09TIPEC	IPEC	EZASOKET API client IVP transaction. *Can be issued directly by a user.*
T09TIPES	IPES	EZASOKET API server IVP transaction. *Can be issued directly by a user.*
T09TCFDM	IPFD	CPT FTP Client Data Manager.
T09TCFDG	IPFG	CPT FTP Client Data Manager for receive.
T09TCFRM	IPFR	CPT FTP Client Response Manager.
T09TLOID	IPFL	Initialization processing transaction.

Program	Trans ID	Description
T09TLCAF	IPLF	Free the LCA control block storage.
T09TTERM	IPPR	Termination transaction. *Can be issued directly by a user*
T09TQUSV	IPQU	Quiesce all active servers. *Can be issued directly by a user.*
T09TSLCT	IPSL	Select Tool transaction.
T09TSTRT	IPST	Initialization transaction. *Can be issued directly by a user.* Must run with TASKDATAKey = CICS.
T09TLSTN	IPTL	Automated listening tool transaction.
T09TRECV	IPTR	Automated receive tool transaction.
T09TSEND	IPTS	Automated send tool transaction.
T09TLST2	IPT2	Automated listening tool transaction phase two.
T09TULST	IPUL	Starts user listeners at CPT initialization time.

Configuring CICS EXCI Communication for CPTMRO

The CPTMRO feature requires that the External CICS Interface be enabled. Use member T09RDOMR as a guide in defining the CICS CSD entries needed to run the CPTMRO

Specify ISC=YES and IRCSTRT=YES in the CICS SIT startup overrides to enable Intersystem Communication for the CICS region.

T09SAMP Member T09RDOMR Defines CPTMRO Entries in the CSD

T09SAMP library member T09RDOMR contains a file stream that can be used to define the necessary CSD entries to CICS for the CPTMRO address space. Sample CSD updates for this can be found in member T09RDOMR. It does the following:

- Removes group T09$EXCI from startup list T09LIST
- Deletes group T09$EXCI
- Define T09CONMR configuration member as a CICS program
- Defines a group T09$EXCI copied from IBM group DFH$EXCI
- Adds alias IPMR to the EXCI transaction with the ALTER command.

  The IPMR transaction must match the TRANSID parameter on the CONNECTION statement in the T09MRO00 configuration statement in the CPTMRO address space
- Adds group T09$EXCI to startup list T09LIST
- Defines CONNECTION IPMR where the NETNAME parameter CPTMRO must match the NETNAME parameter on DEFINE CONNECTION statement in the T09MRO*xx* configuration file
- Defines SESSIONS T09SESS that is related to CONNECTION IPMR
- Lists all the entries in the T09$EXCI group

Member T09RDOMR has question marks in front of statements that make major changes inside your CSD file and causes the statements to fail. Correct the statement using your local settings.

When using T09RDOMR to define your CSD entries for CPTMRO make sure:

- You want to use GROUP T09$EXCI
- You want to use LIST T09LIST
- That the IPMR transaction matches the TRANSID parameter on the CONNECTION statement in the T09MRO00 configuration statement in the CPTMRO address space
- The NETNAME CPTMRO on the DEFINE CONNECTION shown below must match the NETNAME parameter on the CONNECTION statement in the T09MRO00 configuration file in the CPTMRO address space

Sample T09RDOMR Member Entries

```
?REMOVE GROUP(T09$EXCI) LI(T09LIST)
*                                                                     *
?REMOVE GROUP(T09$EXCI) LI(T09LIST)
?DELETE GROUP(T09$EXCI)
*                                                                     *
COPY GROUP(DFH$EXCI) TO(T09$EXCI)
*                                                                     *
ALTER TRANSACTION(EXCI) GROUP(T09$EXCI) ALIAS(IPMR)
*                                                                     *
?ADD  GROUP(T09$EXCI) LI(T09LIST)
*                                                                     *
DEFINE CONNECTION(IPMR) GROUP(T09$EXCI) NETNAME(CPTMRO)                 X
       ACCESSMETHOD(IRC) PROTOCOL(EXCI) CONNTYPE(SPECIFIC)              X
       ATTACHSEC(LOCAL)
*
DEFINE SESSIONS(T09SESS) GROUP(T09$EXCI) CONNECTION(IPMR)               X
       PROTOCOL(EXCI) RECEIVECOUNT(999) RECEIVEPFX(<)
*
LIST GROUP(T09$EXCI) OBJECTS
```

The CPTMRO job defines CICS regions with the DEFINE CONNECTION statement in its T09MRO00 configuration file.

Notice the following entries must be coordinated between T09RDOMR and T09MRO00:

- The DEFINE CONNECTION NETNAME of CPTMRO must match the NETNAME parameter in the DEFINE CONNECTION statement
- The TRANSID IPMR matches the transaction name on the ALTER statement above
- The APPLID must match the CICS APPLID:

```
DEFINE CONNECTION   C1     APPLID        CICSPR01     -
                NETNAME      CPTMRO                   -
               TRANSID       IPMR                     -
               TYPE          SPECIFIC                 -
               LOGLEVEL      WARNING                  -
               LOGLEVEL      DEBUG                    -
               MESSAGELEVEL  WARNING                  -
               MESSAGELEVEL  INFO
```

Note: CPTMRO requires that a site turn on IRC in the CICS SIT override parameters.

To enable IRC inside CICS for use by the CPTMRO address space, place the following entries IRCSTRT=YES into your CICS SIT startup parameters:

```
ISC=YES
IRCSTRT=YES
```

Terminal Control (TCT) Entries

When running a secure command server (All users of the command server are required to provide a valid user ID or password combination and their access to commands is vetted.), you must define a principal facility to associate with the Unicenter NetMaster command processor interface.

Sites running the Command Server without security (parameter SECURITY=N is set on the T09MCMDS macro in the T09CON*xx* configuration file) or without the Command Server Interface (the T09MCMDS macro is not present in the T09CON*xx* configuration file) do not need to perform any actions under this step.

The Unicenter NetMaster command processor interface requires that a principal facility and a terminal be associated with it in order for its security checking to work properly. You can define the terminal using the standard CICS/TS terminal definition macro, DFHTCT. The Unicenter NetMaster command processor never directly references this dummy terminal and it is not used for traditional terminal purposes.

However,

- The terminal name and terminal type you select must be valid as far as CICS/TS is concerned, even though they are meaningless to the Unicenter NetMaster command processor.
- The Unicenter NetMaster command processor is configured by the T09MCMDS macro in the T09CON*xx* configuration member. See the "Configuration Reference" chapter for more information.
- In the T09MCMDS macro, you can select a terminal name using the TERMID= *parameter* or let it default to TCMD. The TERMID= *parameter* of the T09MCMDS macro must match the TRMIDNT= *value* in the DFHTCT entry defining the terminal.

  Note: In the TCT sample below, a field is shown in ***bold italics***. Notice that this TCT entry for ***TCMD*** matches the T09MCMDS default entry.
- You must also include a dummy DD statement for this terminal in your CICS/TS region startup JCL.

  ```
  //PRNT001  DD  DUMMY,DCB=BLKSIZE=80
  ```

  Note: In the following TCT sample, this field is shown in ***bold italics,*** notice that two TCT entries for ***PRNT001*** match the DD statement shown previously.

Sample T09TCT Member

The default distributed sample CICS/TS TCT entries can be copied from the T09SAMP library member T09TCT, which is shown below.

```
TCTJH    TITLE 'DFHTCT MASTER TCT'
         PRINT GEN
         DFHTCT TYPE=INITIAL,                                      X
               ACCMETH=(NONVTAM,VTAM),                             X
               SUFFIX=JH
         DFHTCT TYPE=SDSCI,DEVICE=1403,DSCNAME=PRNT001
         DFHTCT TYPE=LINE,ACCMETH=BSAM,INAREAL=80,TRMTYPE=CRLP,        X
               OSADSCN=PRNT001
         DFHTCT TYPE=TERMINAL,TRMIDNT=TCMD,ERRATT=NO,LPLEN=80,         X
               PGESIZE=(24,80),TRMSTAT=RECEIVE
         DFHTCT TYPE=FINAL
         END   DFHTCTBA
```

Chapter 4 CICS Sockets Compatibility

CICS Sockets Compatibility

The EZACIC01 TRUE processes EZASOKET and EZACICAL calls inside the CICS address space. Unicenter SOLVE:CPT allows sites to configure which (CA's or IBM's) EZACIC01 TRUE implementation will run inside the CICS region. This gives CICS regions the flexibility to run EZASOKET applications over the IBM TCPIP stack at the same time as CPT applications run over CA's TCPaccess TCPIP stack.

IBM's EZACIC01 TRUE is started by either the "EZAO START" transaction or by placing program EZACIC20 inside the PLTPI startup member. IBM's version of EZACIC01 is usually found in the SEZATCP file in the DFHRPL DD. When IBM's EZACIC01 TRUE is enabled Socket Management will only be able to run, monitor and debug CPT applications.

CA EZACIC01 TRUE is started by placing setting EZATRUE=Y on the T09MCICS statement in the T09CONxx configuration file. CA's version EZACIC01 is found in the T09LOAD file in the DFHRPL DD. CA's EZACIC01 TRUE can run EZASOKET or EZACICAL applications over either IBM's TCPIP stack or CA's TCPIP stack. When CA's EZACIC01 TRUE is enabled Socket Management will be able to run, monitor and debug EZASOKET EZACIC01 or CPT applications.

The placement of datasets inside of the DFHRPL DD determines which (IBM's SEZATCP or CA's T09LOAD) EZACIC01 TRUE will be found first when CICS tries to enable the TRUE exit.

Versions of EZASOH03 Supplied by TCPIP Vendors

IBM supplies API module EZASOH03. It is found in the IBM TCPIP SEZALOAD library. IBM's EZASOH03 can only be used to communicate with IBM's TCPIP product.

CA's TCPaccess product also supplies a module called T02PHPNS which has an alias of EZASOH03. It is found in the CA TCPIP LINK library. CA's version of EZASOH03 can only be used to communicate with CA's TCPaccess product.

Thus the placement of the libraries in the STEPLIB is important. A site running IBM's TCPIP cannot use CA's version of EZASOH03 (alias of T02PHPNS) to communicate with IBM's TCPIP product. The first version of EZASOH03 found in a STEPLIB will be the EZASOH03 version available for use by the EZACIC01 TRUE for support of the EZASOKET and EZACIC01.

Parameters for Setting Up CICS Sockets Compatibility

There are two parameters on the T09MCICS macro in the T09CONxx configuration file that defines the EZACIC01 environment to be utilized by Unicenter SOLVE:CPT:

EZATRUE= [<u>Y</u> | N]

APIMODNM=[<u>EZASOH03</u> | T02PHPNS]

The EZATRUE configuration option parameter states whether we will ENABLE and run the EZACIC01 TRUE exit.

CPT will try to enable a TRUE called EZACIC01 when a site sets EZATRUE=Y. EZATRUE =Y is the default. This allows Sockets Management and CPT to run ,debug and monitor EZASOKET, EZACICAL and CPT applications over both IBM's and CA's TCPIP stacks.

When a site sets EZATRUE=N then the EZACIC01 TRUE is available for use by IBM's TCPIP product. Sockets Compatibility to only be able to debug and monitor CPT applications.

The APIMODNM configuration parameter states that API load module is to be loaded at startup (either T02PHPNS or EZASOH03). By default APIMODNM will default to EZASOH03. CA's version of T02PHPNS has an alias of EZASOH03. In cases where IBM's TCPIP stack needs its own version of EZACIC01 and TCPaccess must use its version to handle CPT calls directly into module T02PHPNS.

Sites Running IBM's TCPIP and CA's EZACIC01 TRUE Exit inside CICS

CPT can run over the IBM TCPIP stack inside the CICS address space by utilizing CA's EZACIC01 TRUE. By enabling CA's EZACIC01 TRUE a site can use utilize Sockets Management to run, monitor and help debug CPT, EZASOKET or EZACIC01 applications.

A site running IBM's TCPIP stack can direct EZASOKET and EZACICAL applications over CA's EZACIC01 TRUE by setting the following parameters on the T09MCICS statement in the T09CONxx configuration file:

APIMODNM=EZASOH03

EZATRUE=Y

Ensure the CICS JCL DD concatenations are set as follows:

- Place IBM's SEZALOAD library containing module EZASOH03 in the **STEPLIB** DD.
- Place CA's T09LOAD library containing module EZACIC01 ahead of IBM's SEZATCP library in the **DFHRPL** DD.

Sites Running IBM's TCPIP and IBM's EZACIC01 TRUE Exit inside CICS

A site can reserve IBM's EZACIC01 TRUE for use by the IBM TCPIP stack inside the CICS address space. IBM's TCPIP address space will handle CPT, EZASOKET and EZACICAL calls. Sockets Management will only be able to monitor CPT applications.

IBM's EZACIC01 TRUE is started by either the "EZAO START" transaction or by placing program EZACIC20 inside the PLTPI startup member.

A site running IBM's TCPIP stack can direct EZASOKET and EZACICAL applications calls over IBM's EZACIC01 TRUE by setting the following parameters on the T09MCICS macro in the T09CONxx configuration file:

APIMODNM=EZASOH03

EZATRUE=N

Ensure the CICS JCL DD concatenations are set as follows:

- Place IBM's SEZALOAD library containing module EZASOH03 in the **STEPLIB** DD.
- Place IBM's SEZATCP library containing module EZACIC01 ahead of the CA's T09LOAD library in the **DFHRPL** DD.

Sites Running CA's Unicenter Solve:TCPaccess and CA's EZACIC01 TRUE Exit inside CICS

CPT can run over the CA's TCPIP stack inside the CICS address space by utilizing CA's EZACIC01 TRUE. By enabling CA's EZACIC01 TRUE a site can use utilize Sockets Management to run, monitor and help debug CPT, EZASOKET or EZACIC01 applications.

IBM's EZACIC01 TRUE is started by either the "EZAO START" transaction or by placing program EZACIC20 inside the PLTPI startup member.

A site running CA's Unicenter Solve:TCPaccess TCPIP stack can direct EZASOKET and EZACICAL applications over CA's EZACIC01 TRUE by setting the following parameters on the T09MCICS statement inside the T09CONxx configuration file:

APIMODNM=EZASOH03

EZATRUE=Y

Ensure the CICS JCL DD concatenations are set as follows:

- CA's LINK library containing module EZASOH03 in the **STEPLIB** DD must be found ahead of IBM's SEZALOAD library.
- Place CA's T09LOAD library containing module EZACIC01 ahead of IBM's SEZATCP library in the **DFHRPL** DD.

Sites Running the EZACIC01 TRUE over IBM'S TCPIP While CPT Runs over CA's Unicenter Solve:TCPaccess TCPIP Stack

A site can reserve IBM's EZACIC01 TRUE for use by the IBM TCPIP stack inside the CICS address space. IBM's TCPIP address space will handle EZASOKET and EZACICAL calls while CA's TCPaccess will handle CPT calls. Sockets Compatibility will only be able to monitor CPT applications.

IBM's EZACIC01 TRUE is started by either the "EZAO START" transaction or by placing program EZACIC20 inside the PLTPI startup member.

Sites running IBM's TCPIP can direct EZASOKET and EZACICAL applications calls over the IBM's TCPIP stack and its EZACIC01 TRUE while CPT calls are directed to CA's Unicenter Solve:TCPaccess TCPIP stack by setting the following parameters on the T09MCICS macro inside the T09CONxx configuration file:

APIMODNM=T02PHPNS

EZATRUE=N

Ensure the CICS JCL DD concatenations are set as follows:

- Place IBM's SEZALOAD library containing module EZASOH03 in the **STEPLIB** DD ahead of the CA LINK library containing T02PHPNS.
- Place IBM's SEZATCP library containing module EZACIC01 ahead of the CA's T09LOAD library in the **DFHRPL** DD.

Chapter 5 The CPTMRO Environment

This chapter covers the usage of the CPTMRO feature of Unicenter SOLVE:CPT.

For complete installation instructions, see the "CPTMRO Installation and Configuration" chapter in *Getting Started*.

This chapter discusses the following topics:

- CPTMRO Architecture —Outlines the architecture of the CPTMRO feature
- Commands —Provides an overview on how to use the commands to define and manipulate the CPTMRO environment
- Message/Log Controls —Describes how to control the messages written to the console and logs
- Command Summary Table—Provides a summary table of all the command names and their descriptions
- Command Reference —Describes the syntax and use of commands for defining and controlling the CPTMRO environment
- Sample Definitions—Provides sample definitions
- Sample JCL—Provides sample JCL

The CPTMRO feature extends portions of Unicenter SOLVE:CPT to take advantage of the multiple region operation capability of CICS. CPTMRO allows the Listen tool to run in its own address space and distribute new connections into several CICS regions. In this way, the number of concurrent TCP connections for a given listening port are not restricted to just one CICS. The connections can be distributed over a number of CICS based on selection criteria such as storage, number of sockets, and other CICS information. CPTMRO uses the CICS/ESA 4.1 EXCI feature so CICS must be at a minimum of Release 4.1 and also be configured to run ISC and MRO connections.

CPTMRO Architecture

The CPTMRO architecture is object oriented. CPTMRO consists of objects that deal with specific types of functions. These objects are independent of one another with commands that control their interaction. Operator commands define and control the objects. The goal of the architecture is to make CPTMRO as dynamically configurable and flexible as possible while maintaining a high degree of performance and throughput. Some of the objects require tables from the PARMLIB to control their behavior.

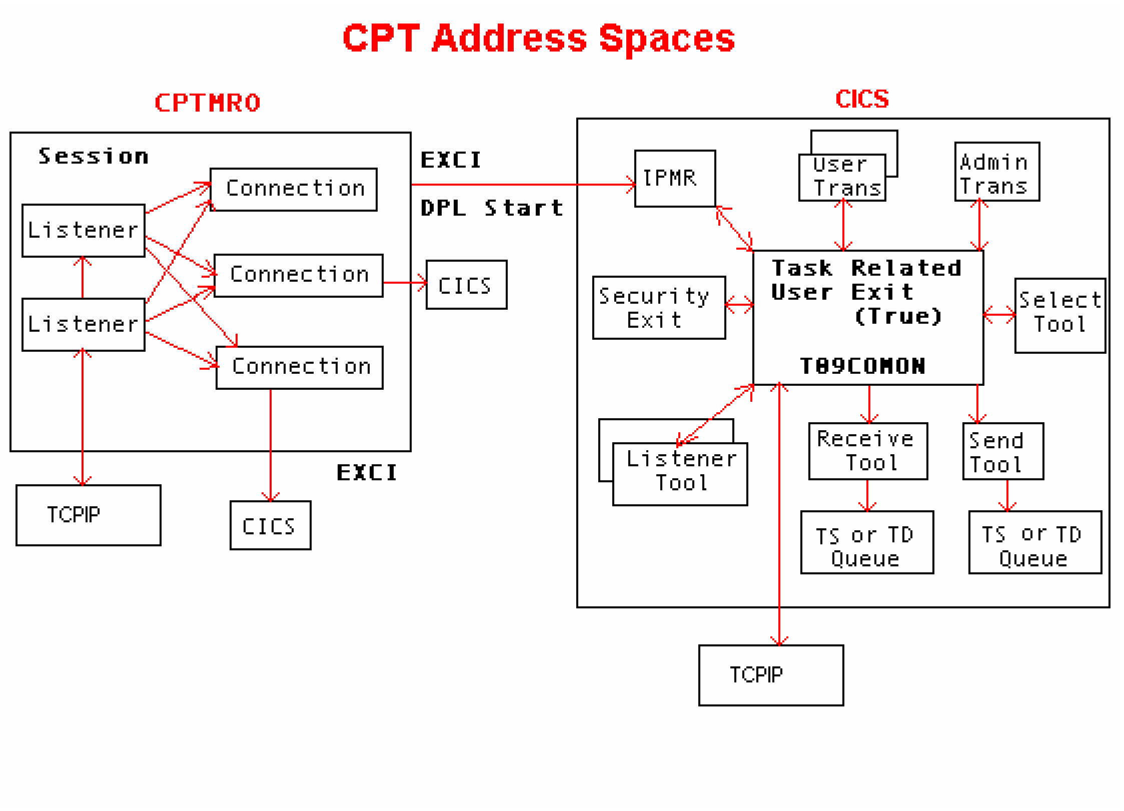
Note: CPTMRO is not IBM CICS MRO compatible. CPTMRO does provide a very similar function as IBM's CICS VTAM based MRO for TCP connections being established into CICS. However, CICS VTAM terminals and TCP connections are inherently different, so the CPTMRO feature mimics this functionality as best as possible.

The major objects that comprise CPTMRO are Listeners, Sessions, Connections, and Log.

- The **Listener** object controls listen objects. Each listen object listens on a different port for remote connections. Once the connection is accepted it is given to a CICS task using a Connection object. The Listener specifies the criteria it wants to use in selecting which CICS task gets the remote connection.
- The **Connection** object controls connection tasks. The connection task is always connected using the EXCI interface to a given CICS. The socket connection is given to the CPT running in the CICS region that starts a transaction to take the socket.
- The **Session** object defines sessions between the listener objects and connection objects. Sessions specify the program that is run in CICS to take the remote connection along with other parameters such as statistics and tracing.
- The **Log** object contains logging functions needed by all the other objects. The log provides messaging and logging to present external information.

Important! *You should always bring up Unicenter SOLVE:CPT in the CICS region first, and then bring up the CPTMRO region. You should also always take down CPTMRO before taking down Unicenter SOLVE:CPT in the CICS region.*

The following diagram provides a pictorial view of the objects and how they relate to one another.



Commands

Commands are the basis for defining and controlling the CPTMRO environment. There is not a separate syntax for defining the environment at startup time versus manipulating the environment during execution. This gives you a consistent way to define and control CPTMRO.

Abbreviation

Command verbs and keywords can be abbreviated to the smallest number of characters that make it unique from the other verbs or keywords. For example, there are the commands CLEAR, DEFINE, EXECUTE, PROMPT, REFRESH, SET, SHOW, SHUTDOWN, START, and STOP. Using the abbreviation technique employed by the parser, CLEAR could be abbreviated to C, CL, CLE, CLEA. DEFINE could be abbreviated as D, DE, DEF, DEFI, DEFIN. However, SET must be abbreviated as SE in order to make it unique from SHOW and SHUTDOWN.

Input Source

The input for the commands can be from any of three sources. The input can be:

- From the console through the MVS MODIFY command interface
- From the console in reply to an outstanding WTOR requested by the PROMPT command
- Read from a member of a PDS pointed to by DDNAME PARMLIB. The member name is a parameter passed on the EXECUTE statement when CPTMRO is started or can be specified on the EXECUTE command

Command Syntax

The commands consist of a command verb, keywords and variable information. Each piece of the command is separated with one or more spaces. Continuation is provided for two of the three sources. For the PROMPT source and EXECUTE source, continuation is signified by a dash (-) as the last character of the input line. CPTMRO recognizes the continuation character and processes the next input line as a continuation of the current input line.

Important! *CPTMRO fully supports mixed case object names. So, for readability within your configuration or an executable PDS member (see the EXECUTE command), this works fine. However, any command entered through a system console is automatically translated to uppercase. This may cause confusing messages, such as object not found. Be aware of this restriction using the console, or use uppercase for all object names.*

Message/Log Controls

CPTMRO produces messages that are sent to the MVS operator console and logged to a designated log.

There are four types of messages as following:

- Error messages
- Informational messages
- Response messages
- Tracing messages

All messages except response messages are controllable through SET commands.

Message Types

Errors

There are three different types of error messages:

FATAL:	Signifies that a catastrophic error occurred and CPTMRO cannot continue.
ERROR:	Indicates that some type of error occurred and action should be taken to correct it.
WARNING:	Signifies that an action did not happen entirely correctly but CPTMRO could make adjustments or use defaults to correct itself.

Informational

There are two types of informational messages: INFO and DEBUG.

INFO:	Produced by CPTMRO to inform the operator about CPTMRO operations and actions in progress.
DEBUG:	Produced to help diagnose problems or provide further detailed information about CPTMRO operations and actions.

Response

Response messages are those messages that CPTMRO always produces in response to a command like a SHOW or SET. The messages always appear so that the operator always knows that CPTMRO recognized the command and has taken action.

Trace

Trace messages only appear in the log. These messages are used to trace events through CPTMRO and should only be used at the direction of Customer Support. The amount of messages generated by trace can be quite large.

Command Summary Table

The command summary table is as follows:

Command	Parameters	Description
CLEAR	*Object_type, object_name*	Removes defined objects from CPTMRO. Alias: DELETE and ERASE
DEFINE CONNECTION	*Object name, parameters*	Creates the configuration of a Connection object.
DEFINE LISTENER	*Object name, parameters*	Creates the configuration of a Listener object.
DEFINE SESSION	*Object name, parameters*	Creates the configuration of a Session object.
EXECUTE	*PDS member name*	Executes a set of commands from a PDS member. Alias: DO, INCLUDE, PERFORM
PROMPT	*None*	Causes a WTOR to be issued, to enter CPTMRO commands against.
REFRESH	*PDS member name*	Activates a new security PDS member name.
SET CONNECTION	*Object name, parameters*	Changes the configuration of a Connection object.

Command	Parameters	Description
SET LISTENER	*Object name, parameters*	Changes the configuration of Listener object.
SET LOG	*Parameters*	Changes log configuration parameters.
SET MRCPT	*Parameters*	Changes the behavior of the main component of CPTMRO.
SET SESSION	*Object name, parameters*	Changes the configuration of Session object.
SHOW CONNECTION	*Object name*	Displays information about connections.
SHOW LISTENER	*Object name*	Displays information about listeners.
SHOW LOG	*None*	Displays information about current LOG settings.
SHOW SESSION	*Object name*	Displays information about sessions.
SHUTDOWN	Type	Terminates the CPTMRO address space.
SPIN	*None*	Closes current logging and create a new log.
START	*Object name*	Makes active (start) the various CPTMRO objects.
STOP	*Object name*	Stops the various CPTMRO objects.

Command Reference

CLEAR

Removes defined objects.

Aliases: DELETE and ERASE.

CLEAR *object name*

where:

object	Type of the object to clear. Choice is LISTENER, SESSION, or CONNECTION.
name	Name of the object to clear. Can be specified as * to clear all names for the specified object.

DEFINE CONNECTION

Creates a Connection object. A Connection represents a connection with a CICS region via the EXCI interface.

The DEFINE CONNECTION command must have at least a name. See the SET CONNECTION command for detailed description of these parameters.

```
DEFINE CONNECTION name
    APPLID appl
    LOGLEVEL     [FATAL | ERROR | WARNING]
               | [INFORMATION | NOINFORMATION | DEBUG | NODEBUG]
    MESSAGELEVEL [FATAL | ERROR | WARNING]
               | [INFORMATION | NOINFORMATION | DEBUG | NODEBUG]
    NETNAME name
    TRACEWORD value
    TRANSID transid | NOTRANSID
    TYPE SPECIFIC | GENERIC
    WAITIME time
```

where:

name	Name for the CONNECTION. It can be up to thirty-two alphanumeric characters.
PARAMETERS	All the parameters listed above are described with the SET CONNECTION command.

DEFINE LISTENER

Creates a Listener object. A Listener represents a process within CPTMRO that listens on a particular port number for incoming connections.

The DEFINE LISTENER command must have at least a name. Other parameters can also be specified. For details about these parameters, see the SET LISTENER command .

```
DEFINE LISTENER name
        LOGLEVEL     [FATAL | ERROR | WARNING] | [INFORMATION | NOINFORMATION |
        DEBUG | NODEBUG]
        MESSAGELEVEL [FATAL | ERROR | WARNING] | [INFORMATION | NOINFORMATION |
        DEBUG | NODEBUG]
        PORT port
        QUEUESIZE number
        RETRYINTERVAL secs
        RETRYMAXIMUM count
        SECURITY member | NOSECURITY
        SELECTMETHOD SOCKETS | TASKS | STORAGE | CIRCULAR | READY
        SYSID sysid
        TRACEWORD value
        WAITTIME time
```

where:

name	Name for the Listener. It can be up to thirty-two alphanumeric characters.
PARAMETERS	All parameters listed above are described with the SET LISTENER command.

DEFINE SESSION

Creates a Session object. A Session represents the path between a Listener object and a Connection object.

The session must have at least a name. Other parameters can also be specified. See the SET SESSION command for details about these parameters.

```
DEFINE SESSION name
        APISTATFLAGS hex-flag
        APITRACEFLAGS hex-flag
        CLIENTLENGTH len
        CLIENTTABLE table | NOCLIENTTABLE
        CLIENTTIME seconds
        CLIENTTRANSLATE | NOCLIENTTRANSLATE
        CONNECTION name
        DESCRIPTION desc | NODESCRIPTION
        DNR | NODNR
        LISTENER name
        LOGLEVEL     [FATAL | ERROR | WARNING] | [INFORMATION | NOINFORMATION |
        DEBUG | NODEBUG]
        MESSAGELEVEL [FATAL | ERROR | WARNING] | [INFORMATION | NOINFORMATION |
        DEBUG | NODEBUG]
        RECEIVECOUNT count
        RECEIVEPARM labelname | NORECEIVEPARM
        RECEIVESIZE size
        SECURITYEXIT program | NOSECURITYEXIT
        SENDCOUNT count
        SENDSIZE size
        SYNCPOINT | NOSYNCPOINT
        TRACEWORD value
        TRANSID tran
        USERID user
```

where:

name	*Sessionname*— can be up to thirty-two alphanumeric characters.
PARAMETERS	The parameters listed above are described with the SET SESSION command.

EXECUTE

Informs CPTMRO to process commands from a member of the PDS referred to by the PARMLIB DD statement in the CPTMRO CPT startup JCL. The command processor reads the PDS member and processes all commands in the member before returning.

Note: The EXECUTE command itself can be in the PDS member but care should be taken to not cause loops with this facility.

Alias: DO, INCLUDE, PERFORM.

```
EXECUTE memname
```

PROMPT

Causes a WTOR to be issued. The normal interface to CPTMRO is with the MVS MODIFY (F) command but this command allows a WTOR interface. While the WTOR is in effect, you can still use the MVS MODIFY command interface.

```
PROMPT           YES | NO
```

where:

YES	Default. Specifies that a WTOR be issued to the console. If a WTOR is already outstanding, then the command has no effect.
NO	Specifies that the WTOR that is outstanding be removed from the console. If there is no WTOR outstanding, an error message is produced.

REFRESH

Refreshes and reloads particular components of the CPTMRO system. The components to refresh are PDS members that were updated and need CPTMRO to reprocess the members to update its internal tables.

```
REFRESH SECURITY memname
```

where:

memname	Name of the PDS member containing the new data for the specified function. The member must reside in the partitioned data set referenced by the PARMLIB DD statement in the startup JCL.

Note: Member must be specified otherwise an error results.

SET CONNECTION

Changes parameter values associated with a Connection object.

Note: These parameters are also valid on a DEFINE CONNECTION command.

```
SET CONNECTION name
    APPLID appl
    LOGLEVEL     [FATAL | ERROR | WARNING]
               | [INFORMATION | NOINFORMATION | DEBUG | NODEBUG]
    MESSAGELEVEL [FATAL | ERROR | WARNING]
               | [INFORMATION | NOINFORMATION | DEBUG | NODEBUG]
    NETNAME name
    TRACEWORD value
    TRANSID transid | NOTRANSID
    TYPE SPECIFIC | GENERIC
    WAITIME time
```

where:

name	Name of the Connection. It can be up to thirty-two alphanumeric characters. It must have been previously defined with a DEFINE CONNECTION command.
PARAMETERS	The above-listed parameters are described in the following sections.

Note: Unless a parameter is specified as dynamic, the Connection must be stopped in order for the parameter to be set.

SET CONNECTION Parameters

This section describes the valid parameters for the SET CONNECTION command.

APPLID

Specifies the generic applid (a one- to eight–byte VTAM applid) of the CICS system to which this Connection object will connect.

```
SET CONNECTION name APPLID appl
```

LOGLEVEL

Controls the types of messages written to the log for the Connection.

Note: This parameter is dynamic.

```
SET CONNECTION name LOGLEVEL [FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]
```

where:

FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Default. Enables warning messaging (implies FATAL and ERROR).
INFORMATION	Default. Enables informational messaging (implies NODEBUG).
NOINFORMATION	Disables informational messaging (implies NODEBUG).
DEBUG	Enables debug messaging (implies INFORMATION).
NODEBUG	Disables debug messaging (implies INFORMATION).

MESSAGELEVEL

Controls the types of messages written to the MVS console through WTO for the Connection.

Note: This parameter is dynamic.

Alias MSGLEVEL.

```
SET CONNECTION name MESSAGELEVEL [FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]
```

where:

FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL). Default.
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION	Default. Enables informational messaging (implies NODEBUG).
NOINFORMATION	Disables informational messaging (implies NODEBUG).
DEBUG	Enables debug messaging (implies INFORMATION).
NODEBUG	Disables debug messaging (implies INFORMATION).

```
SET CONNECTION name NETNAME name
```

where:

name	A one– to eight-byte name.

TRACEWORD

Specifies trace settings for the Connection object.

```
SET CONNECTION name TRACEWORD value
```

where:

value	Trace value to be set in hexadecimal.

TRANSID | NOTRANSID	Specifies the ID of the CICS mirror transaction under which the server program will run. This transaction must be defined to the CICS server region that this Connection is connected with. The NOTRANSID parameter causes the START connection to fail with EXCI response code 12 and reason code 414.

Note: This parameter is dynamic.

```
SET CONNECTION name TRANSID transid
```

where:

transid

One- to four-character ID of the CICS mirror transaction. This transaction must be defined to the CICS server region, and its definition must observe the following rules:

- It must not specify the server program as the initial program of the transaction
- It must specify the mirror program DFHMIRS, and the profile DFHCICSA

Failure to specify DFHMIRS as the initial program means that a COMMAREA passed from the client application program is not passed to the CICS server program. Furthermore, the DPL request fails and the client application program receives a response of SYSTEM_ERROR and reason SERVER_PROTOCOL_ERROR.

The DFHCICSA profile specifies the correct value for the INBFMH parameter, which must be specified as INBFMH(ALL) for a mirror transaction.

The purpose of the *transid* parameter is to distinguish between different invocations of the server program. This enables you to run different invocations of the server program under transactions that specify different attributes. For example, you can vary the transaction priorities, or the security requirements.

TYPE

Specifies the type of Connection with CICS.

```
SET CONNECTION name TYPE SPECIFIC | GENERIC
```

where:

SPECIFIC	The Connection is specific to the CICS and requires a NETNAME be specified on the Connection object. It has a matching CICS CONNECTION definition in the CICS region with the NETNAME and SPECIFIC attributes. **Note:** Default if NETNAME supplied.
GENERIC	The Connection is generic to the CICS. It uses the CICS CONNECTION definition in the CICS region with the GENERIC attribute. **Note:** Default if no NETNAME supplied.

WAITIME

Specifies the amount of time the CONNECTION is idle before issuing an EXCI request to CICS for updated SELECTMETHOD information.

SELECTMETHOD information is always collected after an incoming connection as been given to CICS. This parameter allows the SELECTMETHOD information to be collected even though there were no incoming connections for this CICS for a period of time.

```
SET CONNECTION name WAITIME time
```

where:

time	Number of seconds to wait before sending an inquiry request to CICS for information. Default: 300 seconds.

SET LISTENER

Changes parameter values associated with a listener. These parameters are also valid on a DEFINE LISTENER command.

```
SET LISTENER name
        LOGLEVEL     [FATAL | ERROR | WARNING] | [INFORMATION | NOINFORMATION |
        DEBUG | NODEBUG]
        MESSAGELEVEL [FATAL | ERROR | WARNING] | [INFORMATION | NOINFORMATION |
        DEBUG | NODEBUG]
        PORT port
        QUEUESIZE number
        RETRYINTERVAL secs
        RETRYMAXIMUM count
        SECURITY member | NOSECURITY
        SELECTMETHOD SOCKETS | TASKS | STORAGE | CIRCULAR | READY
        SYSID sysid
        TRACEWORD value
        WAITTIME time
```

where:

name	Name of the listener. It can be up to thirty-two alphanumeric characters. It must be previously defined with a DEFINE LISTENER command.
PARAMETERS	The parameters are described in the following section.

Note: Unless a parameter is specified as dynamic, the listener must be stopped in order for the parameter to be set.

SET LISTENER Parameters

This section describes the valid parameters for the SET LISTENER command.

LOGLEVEL

Controls the types of messages written to the log for the Listener.

Note: This parameter is dynamic.

```
SET LISTENER name LOGLEVEL [FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]
```

where:

FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR). Default.
INFORMATION	Enables informational messaging (implies NODEBUG). Default.
NOINFORMATION	Disables informational messaging (implies NODEBUG).
DEBUG	Enables debug messaging (implies INFORMATION).
NODEBUG	Disables debug messaging (implies INFORMATION).

MESSAGELEVEL

Controls the types of messages written to the MVS console through WTO for the Listener.

Alias MSGLEVEL.

Note: This parameter is dynamic.

```
SET LISTENER name MESSAGELEVEL [FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]
```

where:

FATAL	Enables fatal messaging.
ERROR	Default. Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION	Default. Enables informational messaging (implies NODEBUG).
NOINFORMATION	Disables informational messaging (implies NODEBUG).
DEBUG	Enables debug messaging (implies INFORMATION).
NODEBUG	Disables debug messaging (implies INFORMATION).

PORT

Specifies the port on which the Listener listens. This is the port number to which the remote socket applications will connect.

```
SET LISTENER name PORT port
```

where:

port	Defines the Listener's (server's) well-known port (0-32767).

QUEUESIZE

Specifies the number of connections that can be queued waiting for the listener object to accept or reject the connection. Since the Listener has SOCKETS number of accepts outstanding there should not be a large backup of connection requests. However, if connections are being queued and more connections are queued than QUEUESIZE, the connection is rejected.

```
SET LISTENER name QUEUESIZE number
```

where:

number	Size of the incoming connection queue. Must be numeric. If zero is specified then no connections are allowed. Default 25.

RETRYINTERVAL	Specifies the number of seconds the Listener attempts to restart communication with the TCP provider. Acceptable values are 5 through 86400.

Default: 120 (Two minutes).

```
SET LISTENER name RETRYINTERVAL secs
```

where:

secs	Defines the retry interval in seconds.

RETRYMAXIMUM	Specifies the number of times the Listener attempts to restart communication with the TCP provider.

Acceptable values are 0 through 999. A value of zero means that restart is not to be attempted.

Default: Zero.

```
SET LISTENER name RETRYMAXIMUM count
```

where:

count	Defines the number of times to restart.

SECURITY | NOSECURITY	Specifies a member of the PDS pointed to by the PARMLIB DD JCL statement that describes remote hosts that are allowed access to this Listener.

If this parameter is not specified or is specified as NOSECURITY, then the default is to allow all remote hosts to access the Listener. See the “Security” appendix section CPTMRO Security for a detailed discussion and format of the security member.

```
SET LISTENER name SECURITY member | NOSECURITY
```

where:

member	Name of the PDS member to use.

SELECTMETHOD

Specifies the selection criteria by which the Listener object chooses a Connection object. The Listener chooses a Connection object first by analyzing which Session objects are available and then which Connection objects are available. Then the Listener applies the SELECTMETHOD to the list of Connection objects.

```
SET LISTENER name SELECTMETHOD SOCKETS | TASKS | STORAGE | CIRCULAR | READY
```

where:

SOCKETS	Base the selection on the least number of sockets in use by the CICS region indicated with the CONNECTION parameter.
TASKS	Base the selection on the least number of tasks active in the CICS region indicated with the CONNECTION parameter. The TASKS method causes extra overhead in the CICS region that cannot occur in the SOCKETS, CIRCULAR, or READY methods. If you are routing sessions for this LISTENER to a single CICS region, do not choose the TASKS method.
STORAGE	Base the selection on the least amount of storage in use in the CICS region indicated with the CONNECTION parameter. The STORAGE method causes extra overhead in the CICS region that cannot occur in the SOCKETS, CIRCULAR, or READY methods. If you are routing sessions for this LISTENER to a single CICS region, do not choose the STORAGE method.
CIRCULAR	Rotate to each Session in turn.
READY	Take the first Connection object that is ready to process work.

SYSID

Defines the job name of the TCP/IP provider address space for the Listener.

```
SET LISTENER name SYSID sysid
```

sysid	Defines the TCPIP job name. Default: TCPIP

TRACEWORD	Specifies trace settings for the Listener object.

```
SET LISTENER name TRACEWORD value
```

where:

value	Trace value to be set in hexadecimal.

WAITTIME	Specifies the amount of time to wait for a socket to be taken by the CPT running in the CICS region. If the socket is not taken after this amount of time, it is assumed the transaction was not able to start and the socket is disconnected and closed.

```
SET LISTENER name WAITTIME time
```

where:

time	Number of seconds to wait before closing socket. Default: 30 seconds.

SET LOG

Use the SET LOG command to change parameter values associated with the log. A log is always defined so there is no DEFINE LOG command.

```
SET LOG CLASS class
        FORM form
        DESTINATION dest
        SPINRECORDS recs
        SPINTIME time
        SPINSYNCHRONIZE | NOSPINSYNCHRONIZE
        UPPERCASE | NOUPPERCASE
```

where:

PARAMETERS The parameters are described in the following sections. These parameters can be changed at any time.

SET LOG Parameters

This section describes the valid parameters for the SET LOG command.

CLASS Specifies the JES sysout class that CPTMRO should spool the log. If class is *, the message class (MSGCLASS) for CPTMRO is used as the SYSOUT class.

```
SET LOG CLASS class
```

where:

class	Specifies the desired SYSOUT class. Default: A.

FORM Specifies the JES sysout form that CPTMRO should spool the log.

Note: If not specified the standard default JES form is used.

```
SET LOG FORM form
```

where:

form	Specifies the desired SYSOUT form.

DESTINATION Specifies the JES SYSOUT destination that CPTMRO should spool the log.

Note: If not specified the standard default JES destination is used.

```
SET LOG DESTINATION dest
```

where:

dest	Specifies the desired SYSOUT destination.

SPINRECORDS

Specifies the number of records to be written to the log before it is closed and reopened (spinning). If this parameter is non-zero then log spinning occurs based on record counts even if SPINTIME is specified. When the spin occurs, the currently specified SYSOUT parameters are used to allocate the new log.

```
SET LOG SPINRECORDS recs
```

where:

recs	Number of records to be written to the log. If specified as zero, the log will not be closed and reopened based on records.

Range: 0 to 1000000000 (one billion).

Default: Zero.

SPINTIME

Specifies the time interval in minutes before the log is closed and reopened (spinning). If this parameter is specified, log spinning occurs at the specified time interval even if SPINRECS was just recently reached. When SPINTIME occurs, the count of the number of records is reset to zero. When the spin occurs, the currently specified SYSOUT parameters are used to allocate the new log.

Default: No spinning—occurs based on time interval.

```
SET LOG SPINTIME time
```

where:

time	Specifies in minutes the amount of time that elapses before the log is closed and reopened. Range: 0 to 50400 (five weeks). **Note:** Zero is a special case and is used to specify no spinning to occur based on time.

SPINSYNCHRONIZE | NOSPINSYNCHRONIZE

Indicates, when SPINTIME is in effect, the log close and reopen (spinning) operation occurs synchronized with the time of day (that is, hourly log close/reopen occurs on the hour, daily log close/reopen occurs at midnight, and so on.). Use the NOSPINSYNCHRONIZE parameter to indicate, when SPINTIME is in effect, the log close and reopen (spinning) operation occurs as measured from the start of logging or last spin.

```
SET LOG SPINSYNCHRONIZE
```

```
SET LOG NOSPINSYNCHRONIZE
```

UPPERCASE | NOUPPERCASE

Specifies how messages are written to the CPTMRO log. Messages are normally written in upper and lower case (mixed) but some users may want to upper case the entire message.

Note: Some of the parameters in CPTMRO are specified in lower case and writing messages that contain these parameters to the log in upper case may cause confusion.

```
SET LOG UPPERCASE
```

```
SET LOG NOUPPERCASE
```

where:

SET LOG UPPERCASE	Sets log messages to be written in upper case.
SET LOG NOUPPERCASE	Sets log messages to be written in mixed case.

SET MRCPT

Changes the behavior of the main component of CPTMRO. This component manages all the objects within CPTMRO.

```
SET MRCPT
        LOGLEVEL     [FATAL | ERROR | WARNING] | [INFORMATION | NOINFORMATION |
        DEBUG | NODEBUG]
        MESSAGELEVEL [FATAL | ERROR | WARNING] | [INFORMATION | NOINFORMATION |
        DEBUG | NODEBUG]
        TRACEWORD value
        UPPERCASE | NOUPPERCASE
```

where:

PARAMETERS	The parameters are described in the following section.

SET MRCPT Parameters

This section describes the valid parameters for the SET MRCPT command.

LOGLEVEL

Controls the types of messages written to the log for CPTMRO.

Note: This parameter is dynamic.

```
SET MRCPT LOGLEVEL [FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]
```

where:

FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Default. Enables warning messaging (implies FATAL and ERROR).
INFORMATION	Default. Enables informational messaging (implies NODEBUG).
NOINFORMATION	Disables informational messaging (implies NODEBUG).
DEBUG	Enables debug messaging (implies INFORMATION).
NODEBUG	Disables debug messaging (implies INFORMATION).

MESSAGELEVEL

Controls the types of messages written to the MVS console through WTO for CPTMRO.

Alias MSGLEVEL.

Note: This parameter is dynamic.

```
SET MRCPT MESSAGELEVEL [FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]
```

where:

FATAL	Enables fatal messaging.
ERROR	Default. Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION	Default. Enables informational messaging (implies NODEBUG).
NOINFORMATION	Disables informational messaging (implies NODEBUG).
DEBUG	Enables debug messaging (implies INFORMATION).
NODEBUG	Disables debug messaging (implies INFORMATION).

TRACEWORD

Specifies trace settings for the CPTMRO master task.

```
SET MRCPT name TRACEWORD value
```

where:

value	Trace value to set in hexadecimal.

UPPERCASE | NOUPPERCASE

The UPPERCASE parameter is used to specify how messages are written to the MVS console. Messages are normally written in upper and lower case (mixed) but some users may want to upper case the entire message.

Note: Some of the parameters in CPTMRO are specified in lower case and writing messages that contain these parameters to the console in upper case may cause confusion.

```
SET MRCPT UPPERCASE
```

```
SET MRCPT NOUPPERCASE
```

SET SESSION

Changes parameter values associated with a Session. These parameters are also valid with the DEFINE SESSION command.

```
SET SESSION name
        APISTATFLAGS hex-flag
        APITRACEFLAGS hex-flag
        CLIENTLENGTH len
        CLIENTTABLE table | NOCLIENTTABLE
        CLIENTTIME seconds
        CLIENTTRANSLATE | NOCLIENTTRANSLATE
        CONNECTION name
        DESCRIPTION desc | NODESCRIPTION
        DNR | NODNR
        LISTENER name
        LOGLEVEL     [FATAL | ERROR | WARNING] | [INFORMATION | NOINFORMATION |
        DEBUG | NODEBUG]
        MESSAGELEVEL [FATAL | ERROR | WARNING] | [INFORMATION | NOINFORMATION |
        DEBUG | NODEBUG]
        RECEIVECOUNT count
        RECEIVEPARM labelname | NORECEIVEPARM
        RECEIVESIZE size
        SECURITYEXIT program | NOSECURITYEXIT
        SENDCOUNT count
        SENDSIZE size
        SYNCPOINT | NOSYNCPOINT
        TRACEWORD value
        TRANSID tran
        USERID user
```

where:

name	Name of the Session. It can be up to thirty-two alphanumeric characters. It must be defined with a DEFINE SESSION command.
PARAMETERS	The parameters are described in the following section. Some of the parameters can be set dynamically and others must be set when the connection is stopped. This behavior is noted in each parameter definition.

SET SESSION Parameters

This section describes the valid parameters for the SET SESSION command.

APISTATFLAGS

Sets the API statistics logging options for the CPT application program running in the CICS region. It sets the API statistics flags the same as the ACMSTATS parameter does on a CPT Connect or CPT Listen service call.

Note: This parameter is dynamic.

```
SET SESSION name APISTATFLAGS flag
```

where:

flag	Flag setting in hexadecimal.

APITRACEFLAGS

Important, all tracing is now done via the TCPEEP utility. See the "Diagnostic Commands" chapter for further information. This parameter is now ignored.

Sets the API trace logging options for the CPT application program running in the CICS region. It sets the API trace flags the same as the ACMTRACE parameter does on a CPT Connect or CPT Listen service call.

Note: This parameter is dynamic.

```
SET SESSION name APITRACEFLAGS flag
```

where:

flag	Flag setting in hexadecimal.

CLIENTLENGTH

Specifies the maximum length the Client-Data transaction will receive on its initial receive. The maximum allowed is 50 bytes based on the current Client-Data Option format. The use of this parameter enhances transaction startup performance when the length of the Client-Data is always less than the maximum 50 bytes.

Alias CLNTLEN.

```
SET SESSION name CLIENTLENGTH len
```

where:

len	Maximum number of bytes to receive to determine the next translation. Default: 50.

CLIENTTABLE | NOCLIENTTABLE

Use the CLIENTTABLE parameter with the Client-Data Option (CLIENTTIME specified) to specify the name of the translate table to use when the CLIENTTRANSLATE parameter is specified.

Note: If no translate table is specified, the default translate table is used.

Alias: CLNTTBL.

Default: NOCLIENTTABLE.

```
SET SESSION name CLIENTTABLE table
```

where:

table	Name of the translate table.

CLIENTTIME

Specifies the Client-Data option. With the Client-Data option, CPT running in CICS receives the input stream to determine the transaction ID to start. See Client-Data Listener Option for formats.

Alias CLNTIME.

```
SET SESSION name CLIENTTIME seconds
```

where:

seconds	Number of seconds the Client-Data transaction wait to receive the input data stream which determines the transaction to run from the client.

CLIENTTRANSLATE | NOCLIENTTRANSLATE

Use the CLIENTTRANSLATE parameter with the Client-Data Option (CLIENTTIME specified) to indicate that the initial input stream thatdetermines the next transaction to start is translated.

```
SET SESSION name CLIENTTRANSLATE
```

Alias: CLNTRNS.

Default: NOCLIENTTRANSLATE.

CONNECTION

Associates a Session with a particular Connection object. This specifies one half of the path between a Listener object and a Connection object.

```
SET SESSION name CONNECTION name
```

where:

name	Name of the Connection object.

DESCRIPTION | NODESCRIPTION

Associates an arbitrary value with the Session. It can be used for such things as describing the application this Session represents. The NODESCRIPTION parameter removes a description from the Session object.

Note: This parameter is dynamic.

```
SET SESSION name DESCRIPTION desc
SET SESSION name NODESCRIPTION
```

where:

desc	Description to assign to the Session. Maximum: 32 characters. **Note:** Blank characters are not allowed.

DNR/NODNR

Specifies that the Unicenter SOLVE:CPT code running in CICS perform as DNR lookup for the remote host name based on the remote IP address. If the remote host name is not needed by the application, then it is best to use NODNR for performance reasons.

Note: This parameter is dynamic.

```
SET SESSION name DNR
```

Default is NODNR.

LISTENER

Associates a session with a particular Listener object. This specifies one half of the path between a Listener object and a Connection object.

```
SET SESSION name LISTENER name
```

where:

name	Name of the Listener object.

LOGLEVEL	Controls the types of messages written to the log for the session.

Note: This parameter is dynamic.

```
SET SESSION name LOGLEVEL [FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]
```

where:

FATAL	Enables fatal messaging.
ERROR	Enables error messaging (implies FATAL).
WARNING	Default. Enables warning messaging (implies FATAL and ERROR).
INFORMATION	Default. Enables informational messaging (implies NODEBUG).
NOINFORMATION	Disables informational messaging (implies NODEBUG).
DEBUG	Enables debug messaging (implies INFORMATION).
NODEBUG	Disables debug messaging (implies INFORMATION).

MESSAGELEVEL	Controls the types of messages written to the MVS console through WTO for the Session.

Alias MSGLEVEL.

Note: This parameter is dynamic.

```
SET SESSION name MESSAGELEVEL [FATAL | ERROR | WARNING] |
[INFORMATION | NOINFORMATION | DEBUG | NODEBUG]
```

where:

FATAL	Enables fatal messaging.
ERROR	Default. Enables error messaging (implies FATAL).
WARNING	Enables warning messaging (implies FATAL and ERROR).
INFORMATION	Default. Enables informational messaging (implies NODEBUG).
NOINFORMATION	Disables informational messaging (implies NODEBUG).
DEBUG	Enables debug messaging (implies INFORMATION).
NODEBUG	Disables debug messaging (implies INFORMATION)

RECEIVECOUNT

Specifies the number of input buffers to allocate when data is to be received. These buffers are used to retrieve data from TCP/IP. The number specified here and the RECEIVESIZE value are multiplied to determine total receive storage allocation in CICS for the connection/endpoint.

Alias: RCVCNT.

Note: Any value other than one is ignored. This parameter is dynamic.

```
SET SESSION name RECEIVECOUNT count
```

where:

count	Number of receive buffers. Default is one.

Note: RECEIVECOUNT times RECEIVESIZE must not exceed 61 KB.

RECEIVEPARM | NORECEIVEPARM

Selects the label of a T09MRECV macro defined in the destination CICS CPT configuration. This parameter is used when the destination transaction is the RECEIVE Tool. It is used by the Receive Tool to determine its options and queue name.

User-written applications do not require this parameter. If the label is not defined in the Unicenter SOLVE:CPT configuration macro of the CICS region that the incoming connection is routed to, the connection is terminated and a message is written to the console by the CICS region.

Alias: RCVPARM | NORCVPARM.

```
SET SESSION name RECEIVEPARM rcvparm
```

where:

name	Name of the session being modified.
rcvparm	Label of the CPT Receive Tool in the Unicenter SOLVE:CPT configuration.

RECEIVESIZE

Specifies the size of the receive buffers to allocate when data is to be received. These buffers are used to retrieve data from TCP/IP. The number specified here and the RECEIVECOUNT value are multiplied to determine total receive storage allocation in CICS for the connection/endpoint.

Note: This parameter is dynamic.

Alias: RCVSIZE.

```
SET SESSION name RECEIVESIZE size
```

where:

size	Size in bytes of the receive buffers. Default: 1024.

Note: RECEIVESIZE times RECEIVECOUNT must not exceed 61 KB.

SECURITYEXIT | NOSECURITYEXIT

Specifies the program name of the security exit which should be run in the CICS region before the desired user transaction is started. See the discussion of the Security Exit in the *Programmers Guide* for more information.

Note: This parameter is dynamic.

Alias: SCTYEXIT | NOSCTYEXIT.

Default: NOSECURITYEXIT.

```
SET SESSION name SECURITYEXIT program
```

where:

program	Security program name to use.

SENDCOUNT

Specifies the number of output buffers to allocate when data is to be transmitted. These buffers are used to transmit data to TCP/IP. The number specified here and the SENDSIZE value are multiplied to determine total send storage allocation in CICS for the connection/endpoint.

Alias SNDCNT.

Note: Any value other than one is ignored. This parameter is dynamic.

```
SET SESSION name SENDCOUNT count
```

where:

count	Number of send buffers. Default: One.

Note: SENDCOUNT times SENDSIZE must not exceed 61 KB.

SENDSIZE

Specifies the size of the send buffers to allocate when data is to be transmitted. These buffers are used to transmit data to TCP/IP. The number specified here and the SENDCOUNT value are multiplied to determine total send storage allocation in CICS for the connection/endpoint.

Note: This parameter is dynamic.

Alias: SNDSIZE.

```
SET SESSION name SENDSIZE size
```

where:

size	Size in bytes of the send buffers. Default: 1024.

Note: SENDSIZE times SENDCOUNT must not exceed 61 KB.

SYNCPOINT | NOSYNCPOINT

Use the SYNCPOINT parameter to have the CPT code running in CICS issue a SYNCPOINT before starting the next transaction.

Note: This parameter is dynamic.

Default: NOSYNCPOINT.

```
SET SESSION name SYNCPOINT
```

TRACEWORD

Specifies trace settings for the Session object.

```
SET SESSION name TRACEWORD value
```

where:

value	Trace value to set in hexadecimal.

TRANSID

Specifies the transaction ID to process the TCP connection. This is the transaction to start in the CICS region when the region is determined and the connection is passed.

Note: The TRANSID parameter and the CLIENTTIME parameter are mutually exclusive.

```
SET SESSION name TRANSID tran
```

where:

name	Name of the session being modified.
tran	One- to four-character transaction ID to start.

USERID — Specifies the user ID to use when starting the processing transaction. This lets the started transactions inherit the security permissions of the specified user ID.

```
SET SESSION name USERID user
```

where:

name	Name of the session being modified.
user	*User ID* of the started transaction.

SHOW CONNECTION

Displays information about a particular connection or many connections.

```
SHOW CONNECTION name | *
[CONSOLE | NOCONSOLE] [LOG | NOLOG] [EVERY]
STATUS | SETTINGS | COUNTERS | ALL
```

where:

name	Name of a connection object. Can be specified as * to signify all connections.
STATUS	Displays the status of the connection. Fields to display are name, listener and CICS EXCI task. (Default.)
SETTINGS	Displays all the fields of the connection object as shown in the SET CONNECTION command.
COUNTERS	Displays statistics fields about the connection object such as bytes sent, number of retries, retry wait time, send time, number of jobs sent.

SHOW LISTENER

Displays information about a particular listener or all listeners.

```
SHOW LISTENER name | *
[CONSOLE | NOCONSOLE] [LOG | NOLOG] [EVERY]
STATUS | SETTINGS | COUNTERS | ALL
```

where:

name	Name of a listener object. Can be specified as * to signify all listeners.
STATUS	Default. Displays the status of the listener. Fields to display are name, type, status, sessions (active and total).
SETTINGS	Displays all the fields of the listener object as shown in the SET LISTENER command.
COUNTERS	Displays statistics fields about the listener object such as connections received.

SHOW LOG

Displays the current LOG settings.

```
SHOW LOG
```

The messages displayed by this command show all possible values for the LOG object. All values shown are changeable via the SET LOG command.

SHOW SESSION

Displays information about a particular Session or all Sessions.

```
SHOW SESSION name | *
[CONSOLE | NOCONSOLE] [LOG | NOLOG] [EVERY]
STATUS | SETTINGS | COUNTERS
```

where:

name	Name of a Session object. Can be specified as * to signify all Sessions.
STATUS	Default. Displays the status of the Session. Fields to display are name, type, status, sessions (active and total).
SETTINGS	Displays all the fields of the Session object as shown in the SET Session command.
COUNTERS	Displays statistics fields about the Session object.

SHUTDOWN

Terminates the CPTMRO address space.

```
SHUTDOWN DRAIN | IMMEDIATE | FORCE
```

where:

DRAIN	Default. All Listeners and Connections are quiesced and shutdown is then performed.
IMMEDIATE	Shutdown is done immediately. All functions of the Listeners and Connections are terminated.
FORCE	Shutdown is performed forcibly. CPTMRO does not clean up nor does it close files, it terminates the main program. All object functions are terminated abnormally.

A MVS STOP (P) command may also be used to terminate CPTMRO.
The command is equivalent to the SHUTDOWN IMMEDIATE command.

SPIN

Closes the CPTMRO LOG without stopping the log. This is useful when the log is becoming large and it is desirable to restart the log into either new data set or SYSOUT. This parameter should be used instead of stopping and starting the log since that may cause data to be lost. When the spin occurs, the currently specified SYSOUT parameters are used to allocate the new log.

```
SPIN [LOG]
```

START

Activates the various CPTMRO objects. All CPTMRO objects must be started before use.

START *object name*

where:

object	Type of the object to start. **Note:** Should be LISTENER, SESSION or CONNECTION.
name	Name of the object to start. Can be specified as * to start all startable names for the specified object.

STOP

Inactivates the various CPTMRO objects. When a CPTMRO object is inactive it does not perform any work.

```
STOP object name DRAIN | IMMEDIATE | FORCE
```

where:

object	Type of the object to stop. Should be LISTENER, SESSION or CONNECTION.
name	Name of the object to be stopped. Can be specified as * to stop all names for the specified object.
DRAIN	Default. The object is quiesced and allowed to finish work before becoming inactive.
IMMEDIATE	Object is stopped immediately. All functions of the object are terminated.
FORCE	Object is stopped forcibly. All functions of the object are terminated abnormally. This command can be dangerous. It may leave residual memory allocated by the object. Use with caution. This option is not valid for the connection object.

Sample Definitions

For an explanation of the sample configuration setup for the Installation Verification Procedure (IVP), and a diagram of the interrelationships between the various configuration parameters, see the “CPTMRO Installation and Configuration” chapter in *Getting Started*.

Example

```
*
*  Sample CPT/MRO startup EXEC
*
*----------------------------------------------------------------------
*  Set Global CPT/MRO parameters
*----------------------------------------------------------------------
*
SET    MRCPT                   TRACE         0                         -
                               MESSAGELEVEL  WARN                      -
                               MESSAGELEVEL  NOINFO                    -
                               LOGLEVEL      WARN                      -
                               LOGLEVEL      INFO
*
*----------------------------------------------------------------------
*  Set Log parameters
*----------------------------------------------------------------------
*
*    Log will not be spun based on time or records.  Log messages will
*    be mixed case.
*
*    Edit the following as appropriate:
*
*       CLASS c  - SYSOUT class of the Log dataset
*
SET    LOG                     TRACE         0                         -
                               MESSAGELEVEL  WARN                      -
                               MESSAGELEVEL  INFO                      -
                               LOGLEVEL      WARN                      -
                               LOGLEVEL      INFO                      -
                               CLASS         c                         -
                               SPINREC       0                         -
                               SPINTIME      0                         -
                             NOSPINSYNC                                -
                             NOUPPER
*
*  The first log spin causes logging to become active
*
SPIN   LOG
*
```

```
*----------------------------------------------------------------------
*  Define CPT Listener tasks needed to run the IVP
*----------------------------------------------------------------------
*
*  These Listeners will accept connections on the
*  specified ports and pass them to the CICS region
*  defined on the Connection via an available Session.
*
*  SELECTMETHOD defines the criteria used to choose a
*  Session/Connection/CICS to which to pass the end point.
*
*  Verify PORT and TCPIPJOB meet your local TCP environment
*
DEFINE LISTENER     LISTEN01   PORT          1450                    -
                               TCPIPJOB      TCPIP                   -
                               SELECTMETHOD  READY                   -
                               LOGLEVEL      WARNING                 -
                               LOGLEVEL      DEBUG                   -
                               MESSAGELEVEL  WARNING                 -
                               MESSAGELEVEL  INFO
*
DEFINE LISTENER     LISTEN02   PORT          1451                    -
                               TCPIPJOB      TCPIP                   -
                               SELECTMETHOD  CIRCULAR                -
                               LOGLEVEL      WARNING                 -
                               LOGLEVEL      DEBUG                   -
                               MESSAGELEVEL  WARNING                 -
                               MESSAGELEVEL  INFO
*
DEFINE LISTENER     LISTEN03   PORT          1452                    -
                               TCPIPJOB      TCPIP                   -
                               SELECTMETHOD  SOCKETS                 -
                               LOGLEVEL      WARNING                 -
                               LOGLEVEL      DEBUG                   -
                               MESSAGELEVEL  WARNING                 -
                               MESSAGELEVEL  INFO
*
DEFINE LISTENER     LISTEN04   PORT          1453                    -
                               TCPIPJOB      TCPIP                   -
                               SELECTMETHOD  TASKS                   -
                               LOGLEVEL      WARNING                 -
                               LOGLEVEL      DEBUG                   -
                               MESSAGELEVEL  WARNING                 -
                               MESSAGELEVEL  INFO
*
```

```
*----------------------------------------------------------------------
*  Define a CPT Connection task.
*----------------------------------------------------------------------
*
*  This Connection specifies a CICS region with
*  which to establish an EXCI association.
*
*  TRANSID defines the EXCI mirror transaction defined to CICS.
*  NETNAME is the name used to associate the EXCI client with
*  a CONNECTION definition in CICS of type SPECIFIC
*
DEFINE CONNECTION   CONNECT01  APPLID        CICSxxxx                  -
                               NETNAME       CPTMRO                    -
                               TRANSID       IPMR                      -
                               TYPE          SPECIFIC                  -
                               LOGLEVEL      WARNING                   -
                               LOGLEVEL      DEBUG                     -
                               MESSAGELEVEL  WARNING                   -
                               MESSAGELEVEL  INFO
*
*----------------------------------------------------------------------
*  Define Sessions to associate the Listeners with the specified
*  Connection(s).
*----------------------------------------------------------------------
*
*  RECEIVEPARM is equivalent to the PARM keyword on the T09MLSTN
*  macro in T09CONxx configuration file.
*
DEFINE SESSION      SESS01     LISTENER      LISTEN01                  -
                               CONNECTION    CONNECT01                 -
                               TRANSID       IPTR                      -
                               RECEIVEPARM   IVPRECV1                  -
                               APITRACEFLAGS 0
*
DEFINE SESSION      SESS02     LISTENER      LISTEN02                  -
                               CONNECTION    CONNECT01                 -
                               TRANSID       IPTR                      -
                               RECEIVEPARM   IVPRECV2                  -
                               APITRACEFLAGS 0
*
DEFINE SESSION      SESS03     LISTENER      LISTEN03                  -
                               CONNECTION    CONNECT01                 -
                               TRANSID       IPTR                      -
                               RECEIVEPARM   IVPRECV3                  -
                               APITRACEFLAGS 0
*
DEFINE SESSION      SESS04     LISTENER      LISTEN01                  -
                               CONNECTION    CONNECT01                 -
                               TRANSID       IPTR                      -
                               RECEIVEPARM   IVPRECV4                  -
                               APITRACEFLAGS 0
*
*----------------------------------------------------------------------
*  Start everything
*----------------------------------------------------------------------
*
START LISTENER      *
START CONNECTION    *
START SESSION       * *
```

Sample JCL

For an explanation of the customization of the sample JCL and configuration setup, see the “CPTMRO Installation and Configuration” chapter in *Getting Started*.

For an example of the startup SYSLOG messages, see the IVP for CPTMRO section of the “Installation Verification Procedure (IVP)” chapter.

Important! *To properly initialize, Unicenter SOLVE:CPT must first be started in the CICS region before starting the external CPTMRO address space.*

```
//T09MRO   PROC TRGINDX='CPT.V610',     Hi-level Qual for CPT
//*            TCPLINK='TCP.V610.LINK',  Link Library for TCPaccess
//*            TCPLINK='TCPIP.SEZALINK', Link Library for IBM TCPIP
//             CICSINDX='CICSTS.V220',  Hi-level Qual for CICS/TS
//             USERPARM='&TRGINDX',     DSN for your local parmlib
//             CNFG=T09MRO00,           CPTMRO Config member
//             SOUT='*'                 SYSOUT class
//*
//*        Sample JCL Procedure to run CPTMRO
//*
//T09MRO   EXEC PGM=T09MRCPT,PARM=&CNFG
//STEPLIB  DD DISP=SHR,DSN=&TRGINDX..T09LOAD
//         DD DISP=SHR,DSN=&TRGINDX..SASLINK
//*        DD DISP=SHR,DSN=&TCPLINK
//         DD DISP=SHR,DSN=&CICSINDX..SDFHEXCI
//PARMLIB  DD DISP=SHR,DSN=&USERPARM
//         DD DISP=SHR,DSN=&TRGINDX..T09SAMP
//SYSPRINT DD  SYSOUT=&SOUT
//SYSTERM  DD  SYSOUT=&SOUT
//SYSUDUMP DD  SYSOUT=&SOUT
//SYSTCPD  DD DISP=SHR,DSN=TCPIP.DATA
```

Chapter 6 Operations

This chapter provides operation information for Unicenter SOLVE:CPT.

Important! *All operation information for the CPTMRO feature of Unicenter SOLVE:CPT is contained in the "The CPTMRO Environment" chapter of this guide.*

It includes the following topics:

- Initialization —Describes the methods available for initializing the Unicenter SOLVE:CPT Interface(Interface) and for diagnosing initialization failure
- Termination—Describes the methods available for terminating the Interface and for diagnosing termination failure
- Cumulative Statistics Records—Describes how to collect cumulative statistics of CPT transactions
- Using the IPUL Transaction to Start T09MTRAN Transaction After Product Setup
- Using the IPQU Transaction to Quiesce All Active Servers

The initialization and termination can be executed automatically by using the CICS/TS Program List Tables (PLT). The Interface can be started and stopped manually, any number of times, without the need to bring down the CICS/TS system. This capability to reinitialize the Interface allows a new configuration table module to be loaded at anytime.

Initialization

The Unicenter CPT:SOLVE Interface (the Interface) is initialized by program, T09TSTRT, which has a default transaction ID of IPST. A different transaction ID can be configured in T09MCICS.

Invoke this program using one of the following methods to initialize the Interface:

- Specify the T09TSTRT program as an entry in the CICS/TS PLT table for program initialization (PLTPI). This automatically starts up the Interface as part of CICS/TS initialization. This process is described in the Configuration Table Suffixing section of the “Installation and Configuration” chapter.
- Enter the IPST CP transaction from a CICS/TS terminal. Where, CP is the suffix of your configuration table. This manual start up of the Interface is useful on test systems, where the Interface can be brought up and down manually to test a new Configuration Table. If a new copy of the T09CONCP configuration module was assembled, then the T09TSTRT program automatically loads the newest version of that load module from its DFHRPL load library concatenation. This process is described fully in the Configuration Table Suffixing section of the “Installation and Configuration” chapter.

Regardless of which startup method you use, the T09TSTRT program performs the same set of tasks to initialize the CPT environment:

- Enables the SOLVE:CPT Task-Related User Exit program, T09COMON.
- Enables Task-Related User Exit programs, T09COMON and EZACIC01.
- Loads the Configuration Table module, T09CONFG, or one of the alternative configuration tables of format T09CONCP that is associated with a region, as explained in the Configuration Table Suffixing section of the “Installation and Configuration” chapter.
- Opens the Transient Data queues for Error, Statistics, and Trace (the default names are ACER, ACST, and ACTR, respectively).
- Connects to the address space of the TCP transport provider. The TCP/IP started task name is specified by the JOBNAME= *parm* in the Configuration Table.
- Resolves the local hostname.
- Loads the Translation Table module (the default name is T09XENG).
- Starts all the Listener tools specified in the Configuration Table.
- Starts the Unicenter NetMaster interface command processor if specified in the Configuration Table.
- Starts any user specified listener transactions if specified in the Configuration Table in the T09MTRAN macros.

Unicenter SOLVE:CPT Initialization Messages

When the previous initialization tasks are accomplished, startup messages are written to the system console log. The following set of messages indicates successful initialization:

```
JOB12848  +T09116I T09TSTRT CONFIGURATION TABLE T09CONcp HAS BEEN LOADED
JOB12848  +T09180I T09TSTRT STARTING Unicenter SOLVE:CPT 6.1
JOB12848  +T09183I T09TSTRT LMP Code=ZD,   STARTRAK=SCPT       Abbreviation is Unicenter SOLVE:CPT
JOB12848  +T09100I T09TSTRT CPT TRUE EXIT INTERFACE ENABLEDJOB12848  +T09100I T09TSTRT EZACIC01 TRUE
                   EXIT INTERFACE ENABLED
JOB12848  +T09124I T09CINIT ESTABLISHED SOCKETS COMPATABILITY WITH JOBNAME: tcpip
JOB12848  +T09111I T09CINIT DEFAULT TRANSLATION TABLE T09XENG  LOADED
JOB12848  +T09181I T09TSTRT INITIALIZATION SUCCESSFUL FOR Unicenter SOLVE:CPT 6.1
```

Note: The job name *tcpip* shown in these messages is the default started task name for the transport provider, and may be different for your installation.

A set of messages is also written to the Unicenter SOLVE:CPT Transient Data queues at initialization. In most cases, this output goes to the MSGUSR DD; otherwise, the messages go to wherever the TDQ ACER is redirected within the CICS/TS region. If all three queues are routed to the same CICS/TS SYSOUT data set, then these messages appear together in the CICS/TS SYSOUT to indicate successful initialization:

```
00029 19:02:51 T09123I T09CINIT INITIAL WRITE TO ERROR LOG TD QUEUE
00029 19:02:51 T09810I T09CINIT INITIAL WRITE TO STATISTICS LOG TD QUEUE
00029 19:02:51 T09936I T09CINIT INITIAL WRITE TO TRACE LOG TD QUEUE
00029 19:02:51 T09124I T09CINIT ESTABLISHED SOCKETS COMPATIBILITY WITH JOBNAME: tcpip
00029 19:02:51 T09115I T09CINIT Local home IP address resolved to 141.202.200.70
00029 19:02:51 T09111I T09CINIT DEFAULT TRANSLATION TABLE T09XENG  LOADED
```

The 00029 preceding these messages is the CICS/TS task number of the IPST startup transaction performing the initialization.

Note: The T09115I message verifies whether the IP address is valid for use with your TCPIP job and CICS region. If the xxx.xxx.xxx.xxx IP address is incorrect sites should examine the following for errors:

- The JOBNAME parameter on the T09MCICS statement in configuration file T09CONxx.
- The SYSTCPD DD file in the CICS startup JCL.

Initialization Failure

If a Unicenter SOLVE:CPT/Tool transaction or a user-written application using the product's services gets an AEY9 abend, it is an indication that:

- The Interface was not started
- The initialization failed

Note: AEY9 is a standard CICS/TS abend issued when an application call is made to a task-related user exit that is not enabled.

There are several reasons why Unicenter SOLVE:CPT initialization might fail. First, the TCP address space must be up and running with its API active when Unicenter SOLVE:CPT initialization is started. It must be the TCP transport provider specified by the JOBNAME= *parm* of the current configuration table. If the TCP API is not available, initialization fails and the following messages are sent to the console log:

```
+T09116I T09TSTRT CONFIGURATION TABLE T09CONcp HAS BEEN LOADED
+T09180I T09TSTRT STARTING Unicenter SOLVE:CPT 6.1
+T09183I T09TSTRT LMP Code=ZD, STARTRAK=SCPT Abbreviation is Unicenter TCPaccess
+T09100I T09TSTRT CPT TRUE EXIT INTERFACE ENABLED
+T09100I T09TSTRT EZACIC01 TRUE EXIT INTERFACE ENABLED
+T09102I T09CINIT PRODUCT INITIALIZATION PENDING API CONNECTION
+T09129E T09CINIT UNABLE TO INITIALIZE API FOR JOBNAME tcpip, ERRNO=10102
+T09105E T09CINIT INITIALIZATION FAILED - CHECK CICS LOG FOR DETAILS
+T09114E T09TSTRT INITIALIZATION ROUTINE FAILED RC=  18
+T09132E T09TSTRT CPT TRUE EXIT INTERFACE DISABLED
+T09132E T09TSTRT EZACIC01 TRUE EXIT INTERFACE DISABLED
+T09076I T09TSTRT RESTART HAS BEEN SCHEDULED
```

The *tcpip* shown in message T09129E is the JOBNAME= *parm* in the T09MCICS configuration table entry.

Another reason for initialization failure is if the Task-Related User Exit (TRUE) cannot be enabled. This occurs if CICS/TS cannot load the TRU Exit modules, T09COMON or EZACIC01, or the configuration table module, T09CON*cp*, or the translation table module, T09XENG, or user-provided alternate translation table. This can occur if the actual load module is missing from a DFHRPL load library, or if a problem exists with its CICS/TS RDO definition, such as having the module currently marked DISABLED.

Finally, initialization can fail if any of the three Transient Data queues specified by the QNAMES= *parm* in the configuration table are not defined in the CICS/TS RDO definitions or cannot be opened. In all of these cases, error messages are sent to the console syslog, ending with the T09105E…INITIALIZATION FAILED message.

Unicenter SOLVE:CPT Termination

The Interface is terminated by program, T09TTERM, whose default transaction ID is IPPR. This program can be invoked in the following ways to shutdown the Interface:

- Specify the T09TTERM program as an entry in the CICS/TS PLT table for system shutdown (PLTSD)

  This automatically shuts down the Interface as part of CICS/TS termination.

  This process is described in the Configuration Table Suffixing section of the chapter "T09CON*xx* Customization"
- Enter the IPPR transaction from a CICS/TS terminal

  This manual shutdown of the Interface is useful on test systems, perhaps to load a new copy of the configuration table module, T09CONcp
- Enter the CPTbounce command from the Unicenter NetMaster command processor interface

Regardless of how the shutdown program, T09TTERM, is invoked, it performs the same set of tasks to terminate the Unicenter SOLVE:CPT environment:

- Issues an abortive close on all TCP and UDP endpoints still active
- Disconnects from the address space of the TCP transport provider, whose *tcpip* started task name is specified by the JOBNAME= *parm* in the configuration table
- Releases the Translation Table module, T09XENG
- Disables the Task-Related User Exit programs, T09COMON and EZACIC01
- Releases the configuration table module, T09CON*cp*

Termination Messages

As these tasks are accomplished, termination messages are written to the system console log.

The following messages indicate a successful termination:

```
15.02.05 JOB26273  +T09182I T09CTERM SHUTDOWN COMPLETE FOR Unicenter NetMaster Socket Management for CICS
                        1.1
15.02.05 JOB26273  +T09103I T09CTERM CPT TERMINATED API SUBSYSTEM SESSION
15.02.05 JOB26273  +T09182I T09CTERM SHUTDOWN COMPLETE FOR Unicenter SOLVE:CPT 6.1
15.02.05 JOB26273  +T09132I T09TTERM CPT TRUE EXIT INTERFACE DISABLED
15.02.05 JOB26273  +T09132I T09TTERM T09CIC01 TRUE EXIT INTERFACE DISABLED
15.02.05 JOB26273  +T09132I T09TTERM EZACIC01 TRUE EXIT INTERFACE DISABLED
15.02.05 JOB26273  +T09120I T09TTERM TERMINATION SUCCESSFUL
```

There is also a set of messages written to the Transient Data queues at CPT termination. In most cases, this output goes to the MSGUSR DD, otherwise the messages go wherever the TDQ ACER is redirected to within the CICS/TS region.

Note: Read the notes intermixed with the messages below to learn which of these messages to expect in your environment. The following example shows a subset of messages that would be expected if you where running the default IVP configuration file, T09CONCP.

The following messages show the initial statistics message for all four of the default IVP configuration: 1350-1353. These will change, depending on which listeners you configure for your environment.

```
00247 15:02:02 T09809I API LISTEN STATS - APIOPN= 1      TRANS=0      BAD TRANS=0      TREJECT=0
00247 15:02:02 T09660I T09ESTAT Server  1353  Token=00000009  Mother Session Statistics
00246 15:02:02 T09809I API LISTEN STATS - APIOPN= 1      TRANS=0      BAD TRANS=0      TREJECT=0
00246 15:02:02 T09660I T09ESTAT Server  1352  Token=0000000B  Mother Session Statistics
00245 15:02:02 T09809I API LISTEN STATS - APIOPN= 1      TRANS=0      BAD TRANS=0      TREJECT=0
00245 15:02:02 T09660I T09ESTAT Server  1351  Token=0000000D  Mother Session Statistics
00244 15:02:02 T09809I API LISTEN STATS - APIOPN= 1      TRANS=0      BAD TRANS=0      TREJECT=0
00244 15:02:02 T09660I T09ESTAT Server  1350  Token=0000000F  Mother Session Statistics
```

This message appears if you are running the Unicenter NetMaster Interface:

```
00249 15:02:02 T09495I T09TCMDS NetMaster command server terminating
```

The following example shows the statistics for the server on port 3022. There is a set of these messages for each active listener running in your environment.

```
00509 17:29:27 T09660I T09ESTAT Server  3022  Token=00000011  Mother Session Statistics
00509 17:29:27 T09664I T09ESTAT Server  3022       EZASOKET Verb       Calls  Error  Response Time average
00509 17:29:27 T09665I T09ESTAT Server  3022       ACCEPT             00000019  0000  Seconds 0.001222
00509 17:29:27 T09665I T09ESTAT Server  3022       BIND               00000001  0000  Seconds 0.000166
00509 17:29:27 T09665I T09ESTAT Server  3022       CLOSE              00000001  0000  Seconds 0.000314
00509 17:29:27 T09665I T09ESTAT Server  3022       GETCLIENTID        00000001  0000  Seconds 0.000046
00509 17:29:27 T09665I T09ESTAT Server  3022       INITAPI            00000001  0000  Seconds 0.001428
00509 17:29:27 T09665I T09ESTAT Server  3022       LISTEN             00000001  0000  Seconds 0.000148
00509 17:29:27 T09665I T09ESTAT Server  3022       SELECT             00000032  0000  Seconds 14.659295
00509 17:29:27 T09665I T09ESTAT Server  3022       SOCKET             00000001  0000  Seconds 0.000762
00509 17:29:27 T09663I T09ESTAT Session 3022  25 Daughter Sessions
00509 17:29:27 T09661I T09ESTAT Session 3022  Sent: 00000000 00004708  #Send=382  AvgSess=727  AvgSend=47
00509 17:29:27 T09662I T09ESTAT Session 3022  Recv: 00000000 000029C6  #Recv=407  AvgSess=427  AvgRecv=26
00509 17:29:27 T09664I T09ESTAT Session 3022       EZASOKET Verb       Calls  Error  Response Time average
00509 17:29:27 T09665I T09ESTAT Session 3022       CLOSE              000001C9  0000  Seconds 0.000070
00509 17:29:27 T09665I T09ESTAT Session 3022       GETCLIENTID        000001B0  0000  Seconds 0.000039
00509 17:29:27 T09665I T09ESTAT Session 3022       GETPEERNAME        00000019  0000  Seconds 0.000042
00509 17:29:27 T09665I T09ESTAT Session 3022       GIVESOCKET         000001B0  0000  Seconds 0.000063
00509 17:29:27 T09665I T09ESTAT Session 3022       READ               00000019  0000  Seconds 0.000055
00509 17:29:27 T09665I T09ESTAT Session 3022       RECVFROM           0000017E  0000  Seconds 1.958781
00509 17:29:27 T09665I T09ESTAT Session 3022       SELECT             000001B0  0000  Seconds 0.003183
00509 17:29:27 T09665I T09ESTAT Session 3022       SEND               00000019  0000  Seconds 0.000103
00509 17:29:27 T09665I T09ESTAT Session 3022       TAKESOCKET         000001B0  0000  Seconds 0.000120
00509 17:29:27 T09665I T09ESTAT Session 3022       WRITE              00000165  0000  Seconds 0.000105
```

The following messages displays for each of four default IVP configuration: 1350-1353. You will see one of these messages for each active listener you configure for your environment that is shutdown with Unicenter SOLVE:CPT.

```
00274 15:02:04 T09106I T09CTERM API ENDPOINT 15CAE508 CLOSED
00274 15:02:04 T09106I T09CTERM API ENDPOINT 15CC44D8 CLOSED
00274 15:02:04 T09106I T09CTERM API ENDPOINT 15CC44D8 CLOSED
00274 15:02:04 T09106I T09CTERM API ENDPOINT 15CD0308 CLOSED
```

The following message block always displays:

```
00274 15:02:05 T09103I T09CTERM CPT TERMINATED API SUBSYSTEM SESSION
00274 15:02:05 T09130I T09TTERM PROGRAMMER INTERFACE DISABLED
00274 15:02:05 T09131I T09TTERM T09CONxx CONFIGURATION TABLE RELEASED
```

The 002*nn* preceding these messages is the CICS/TS task number of the CICS/TS transaction on whose behalf these messages were issued.

TCP Transport Provider Failure

If the address space for the TCP transport provider fails or is shut down, the Unicenter SOLVE:CPT termination program, T09TTERM, is automatically invoked. All application endpoints are abortively closed, as if the IPPR transaction had been executed. The messages sent to the console log and TD queues are the same as when the termination program is invoked by transaction or the Administrator Interface.

It is the responsibility of the application program to handle the return codes for the immediate disconnects that could be returned from its last Unicenter SOLVE:CPT call indicating that the Interface is terminating. The application should rollback any committed resources if it has an established syncpoint. The Unicenter SOLVE:CPT Tools do this automatically when their Transient Data queues are protected. The application should also be ready to handle the AEY9 abend that occurs if the application tries to issue a Unicenter SOLVE:CPT call when the Interface is down.

Cumulative Statistics Records

The Administrator Interface can collect cumulative statistics on the Client or Server ports used by CICS transactions.

These statistics have counts for:

- The total number of connections made to that local or remote port
- The total number of messages sent and received on that port
- The cumulative byte length of all those messages
- A breakout of all the socket calls used

These counts can show the volume of activity for a particular CPT application. The earlier example in the Termination Messages section is also shown here.

```
                         00509 17:29:27 T09660I T09ESTAT Server  3022  Token=00000011  Mother Session
                         Statistics
00509 17:29:27 T09664I T09ESTAT Server  3022       EZASOKET Verb      Calls  Error  Response Time average
00509 17:29:27 T09665I T09ESTAT Server  3022       ACCEPT           00000019  0000  Seconds 0.001222
00509 17:29:27 T09665I T09ESTAT Server  3022       BIND             00000001  0000  Seconds 0.000166
00509 17:29:27 T09665I T09ESTAT Server  3022       CLOSE            00000001  0000  Seconds 0.000314
00509 17:29:27 T09665I T09ESTAT Server  3022       GETCLIENTID      00000001  0000  Seconds 0.000046
00509 17:29:27 T09665I T09ESTAT Server  3022       INITAPI          00000001  0000  Seconds 0.001428
00509 17:29:27 T09665I T09ESTAT Server  3022       LISTEN           00000001  0000  Seconds 0.000148
00509 17:29:27 T09665I T09ESTAT Server  3022       SELECT           00000032  0000  Seconds 14.659295
00509 17:29:27 T09665I T09ESTAT Server  3022       SOCKET           00000001  0000  Seconds 0.000762
00509 17:29:27 T09663I T09ESTAT Session 3022  25 Daughter Sessions
00509 17:29:27 T09661I T09ESTAT Session 3022  Sent: 00000000 00004708  #Send=382  AvgSess=727  AvgSend=47
00509 17:29:27 T09662I T09ESTAT Session 3022  Recv: 00000000 000029C6  #Recv=407  AvgSess=427  AvgRecv=26
00509 17:29:27 T09664I T09ESTAT Session 3022       EZASOKET Verb      Calls  Error  Response Time average
00509 17:29:27 T09665I T09ESTAT Session 3022       CLOSE            000001C9  0000  Seconds 0.000070
00509 17:29:27 T09665I T09ESTAT Session 3022       GETCLIENTID      000001B0  0000  Seconds 0.000039
00509 17:29:27 T09665I T09ESTAT Session 3022       GETPEERNAME      00000019  0000  Seconds 0.000042
00509 17:29:27 T09665I T09ESTAT Session 3022       GIVESOCKET       000001B0  0000  Seconds 0.000063
00509 17:29:27 T09665I T09ESTAT Session 3022       READ             00000019  0000  Seconds 0.000055
00509 17:29:27 T09665I T09ESTAT Session 3022       RECVFROM         0000017E  0000  Seconds 1.958781
00509 17:29:27 T09665I T09ESTAT Session 3022       SELECT           000001B0  0000  Seconds 0.003183
00509 17:29:27 T09665I T09ESTAT Session 3022       SEND             00000019  0000  Seconds 0.000103
00509 17:29:27 T09665I T09ESTAT Session 3022       TAKESOCKET       000001B0  0000  Seconds 0.000120
00509 17:29:27 T09665I T09ESTAT Session 3022       WRITE            00000165  0000  Seconds 0.000105
```

These statistical counts are normally displayed on the Browse Detail panels of the Administrator Interface, but they can also be collected for offline batch analysis. To do this, the AITDSTAT= parm must be specified on the T09MCICS macro in the Configuration Table. This parm specifies the name of the Transient Data queue that receives the cumulative statistics records when they are *discarded* from the Administrator Interface.

The cumulative statistics can be reset at any time in the Administrator Interface by using the ZS function on the Operator Control Menu. If the Statistics Discard queue was defined, then the currently accumulated statistics are written out as a separate TD records to this queue before reinitializing the counts. When the Interface is terminated, these records are written out again to the queue, so that a history of total activity can be maintained.

The cumulative statistics records are not in a printable format, so this queue should not be routed to a SYSOUT data set, like the three standard CPT logs. Instead, these records should be sent to a sequential data set, where they can be collected and later analyzed by a batch program.

Note: If you do not intend to do any historical or cumulative analysis of these records, then it is best not to define the Statistics Discard queue at all. Just let the AITDSTAT= parm default to a null queue name, which means that the counts will be discarded when the statistics are reset or Unicenter SOLVE:CPT is shutdown.

If you do intend to write a batch analysis program, there is an Assembly language description of the fields in member T09AITAB in the Unicenter SOLVE:CPT macro library.

Using the IPUL Transaction to Start T09MTRAN Transactions After Product Startup

Any time while the product is running a user can use the IPUL transaction (The transaction specified on the USRTRNID parameter of the T09MCICS statement and program T09TULST.) to start any of the T09MTRAN transactions when passed the correct TRANSID and optionally the ID parameter.

The IPUL transaction starts a transaction from the T09MTRAN chain of entries. There are two positional parameters:

- The mandatory TRANSID (for transaction name)
- The optional ID (for identification).

Each T09MTRAN must have a unique ID while the TRANSID is not required to be unique. T09MTRAN will generate a unique TRANSID if you let it.

The following are sample T09MTRAN entries for a T09CON*xx* configuration file:

```
T09MTRAN TRANSID=SRV1,PARM=1344

T09MTRAN TRANSID=SRV2

T09MTRAN TRANSID=SRV3,PARM='1346,IP=138.141.222.17',ID=ID1346

T09MTRAN TRANSID=SRV3,PARM=1347,ID=ID1347

T09MTRAN TRANSID=SRV4,PORT=4444
```

To start transaction SRV1, you can issue the following command and transaction SRV1 will start with ‘1344’ passed as character data:

```
IPUL SRV1
```

To start transaction SRV2 where no data is passed to the application you can issue the following command:

```
IPUL SRV2
```

To start the correct SRV3 transaction you should issue the IPUL transaction with both of the TRANSID and ID parameters specified:

```
IPUL SRV3 ID1346
IPUL SRV3 ID1347
```

An IPUL SRV3 command would just start the first SRV3 entry (ID1346) in the T09CON*xx* configuration file.

To start transaction SRV4 with a the LSTP DSECT as the passed parameter (because PORT is a CFG0000 parameter on the T09MTRAN macro) specify:

```
IPUL SRV4
```

Using the IPQU Transaction to Quiesce All Active Servers

A site can use the IPQU transaction to quiesce all active server sessions. The servers will no longer process new inbound sessions after the IPQU transaction. It could be entered before operations use the IPPR transaction.

The IPQU transaction takes no parameters when entered:

IPQU

A site should make sure that the IPQU transaction is only available to personnel that require the ability to shut down all active servers in a CICS region.

Chapter 7 Diagnostic Commands

This chapter describes the diagnostic commands available.

It includes the following topics:

- TCPEEP—A command to invoke the packet trace program and diagnose remote host communication problems
- TRACE—Trace command enhancements to collect TCP/IP data and display it on a terminal or send it to an external writer
- Tracing TCP and Socket calls over IBM's TCPIP Stack—Trace command enhancements to collect TCP/IP data and display
- MVS DUMP and SLIP SET commands—Discussion regarding how to capture the IFS TRACE address space using these MVS Console commands

TCPEEP

TCPEEP is a TSO command that invokes the packet trace program to diagnose remote host communication problems. The TCPEEP real-time trace consists of selected network packet traffic to and from a local host. The TCPEEP command recognizes many levels of network traffic, however, we are only addressing CICS/TS socket tracing in this chapter.

TCPEEP creates a NO WRAP Component Trace Instance and displays the output on a TSO terminal or directs it to a dynamically allocated SYSOUT data set. Optionally, it can stop any component trace instance, or modify an existing component trace instance or view an existing Component Trace Instance.

Note: TCPEEP runs only when the TRACE address space is active.

User Interface

TCPEEP can be run as a TSO command, either from TSO or as a batch TSO.

The JCL to run TCPEEP as a batch job can be copied from the T09SAMP library, member T09PEEP. TRACE must be up and running before submitting a batch job for TCPEEP.

The following is sample JCL for running TCPEEP in batch.

```
//T09PEEP  JOB (TCPEEP),'TCPEEP',CLASS=A,MSGCLASS=X
//*
//*  SAMPLE JCL TO RUN TCPEEP IN BATCH.
//*
//*  UPDATE 'TRGINDX' TO REFLECT YOUR LIBRARY NAMING CONVENTION.
//*
//*  NOTE: THE TRACE ADDRESS SPACES MUST BE RUNNING.
//*
//*  NOTE: IF UNICENTER TCPACCESS, THEN IT TOO MUST BE RUNNING.
//*
//TCPEEP   EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=4M
//STEPLIB  DD DISP=SHR,DSN=TRGINDX.LINK
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*,LRECL=132,BLKSIZE=132,RECFM=FB
//SYSIN    DD DUMMY
//*
//* USE THE FOLLOWING TCPEEP FOR UNICENTER TCPACCESS INSTALLATIONS
//*
//SYSTSIN  DD *
 TCPEEP TRCSSID(ACTR) +
 FORMAT(TCP,DATA(16),IUCVDATA(16),EZADATA(16),CPTDATA(16)) +
 GROUPS((NETIF,'PROTOCOL(TCP),MDATA(9999)'), +
        (EZA,'MAXEZADATA(16)'),   +
        (CPT,'MCDATA(16)'),       +
        (IUCV,'MAXIUCVDATA(16)'))
//
//
//* IUCV and TCP parameters are not available for use by IBM TCP/IP
//* USE THE FOLLOWING TCPEEP FOR IBM TCP/IP INSTALLATIONS
//*
//*SYSTSIN  DD *
 TCPEEP TRCSSID(ACTR) FORMAT(EZADATA(16),CPTDATA(16)) +
 GROUPS((EZA,'MAXEZADATA(16)'), +
        (CPT,'MCDATA(16)'))
/*
```

To stop the TCPEEP batch job, issue the MVS **STOP** command (for example, **P** *jobname*).

Trace Data Collected

The trace facility collects the following type of trace data:

- Calls and data moving in and out of an application
- EZASOCKET and EZACICAL calls between Unicenter SOLVE:CPT and the TCP/IP stack
- Optionally, HPNS and Unicenter SOLVE:CPT calls from the TCP/IP address space, if you are using Unicenter TCPaccess

Simultaneous tracing of various trace types can be fed into a single trace output.

Note: If data is collected via the external writer to an external data set, you must use the MVS TRACE command. See External Writer for more information.

Viewing Trace Data

TCPEEP collects real-time data and by default writes to SYSTSPRT, your TSO screen. Optionally, it can write to a dynamically allocated SYSOUT data set, so that it can be viewed in the JES spool.

TRACE Operation

Each occurrence of the TCPEEP command varies according by environment.

To stop a trace, press the terminal ATTENTION key and enter **H** at the prompt (null entry allows trace to continue).

WARNING! *TCPEEP should be installed in a protected library only. It can be used to display all network traffic through Unicenter SOLVE:CPT , and Unicenter TCPaccess, including user IDs, passwords, or even payroll information if it goes out on a network.*

TCPEEP Syntax

This section describes the TCPEEP syntax and describes its parameters.

```
TCPEEP [ ASID( asid,... ) ][ BUFFERS ( size, number ) ][ BUFFTIME ( time_out ) ]
       [ DATASIZE( record_size )] [ FORMAT( format_options ) ]
       [ FULL | SUMMARY ][ GROUPS( ( group [,'filter' ) ... ) ][ HALT ]
       [ INSTANCE ( instance_ID ) ][ JOBNAME ( jobname,... ) ][ NOHEADER ]
       [ PEEK ( limit ) ][ SYSOUT ( class ) ][ TRACESIZE ( num_records ) ]
       [ TRCSSID ( ssid ) ]
```

Note: Also refer to the “Diagnostic Commands” chapter of *Unicenter TCPaccess Communications Server System Management Guide* for other TCPEEP options that can be coded together with the Socket Management options.

ASID (*asid* ,...)	Specifies the address space identifiers (ASIDs) of address spaces used as a filter for tracing. Events in the ASIDs are recorded by the component trace. The parameter contains a list of 0 to 16 hexadecimal ASIDs separated by commas. An empty ASID list, ASID=(), turns off filtering by address spaces. In the ASID parameter, list all address spaces to trace. Address spaces for previous traces are not traced unless listed. Default: None.
BUFFERS (*size*,*number*)	Specifies the size of the trace buffers in kilobytes or the number of buffers.

size	A value between 64 and 1024. Default: 256.
num	A value between 2 and 128. Default: Four.
BUFFERS	(Optional). Can only be specified when creating a new trace instance. If specified when modifying a trace instance, it is ignored.

BUFFTIME (*time_out*)	Specifies the buffer time out interval in seconds. At the end of each interval, if the current buffer contains data but is not full, a buffer flush operation is initiated. BUFFTIME is optional and can only be specified when creating a trace instance. If specified when modifying a trace instance, it is ignored. Use this parameter to force a buffer switch so you do not have to wait for the entire buffer to fill to see trace data. Range: 0 – 99999. Default: 10.

DATASIZE (*record_size*)

Specifies the maximum size of a trace record in kilobytes. Trace records that exceed the specified value are truncated.

DATASIZE—(Optional). Can only be specified when creating a trace instance. If specified when modifying a trace instance, it is ignored.

If the specified maximum size exceeds the largest supported trace record size (64 KB less control headers), then the specification has no effect.

Default: None.

FORMAT (*format_options*)

The format of the collected data.

Formatted data CPT trace records can be specified as follows:

CPTDATA | CDATA | CPTEBCDIC | CPTASCII
or
CPTDATA(*nn*) | CDATA(*nn*) | CPTEBCDIC(*nn*) | CPTASCII(*nn*)

- Where *nn* is between 0 and 65535 (default is 65535)

  Note: A data amount set very high generates a very large trace in a short period. A high value should be avoided unless you are absolutely convinced that there is a data integrity problem. For the vast majority of problems a glance value of 16 is sufficient. A value of 16 causes one line of data to displays.

- CPTDATA | CDATA uses a mixed ASCII and EBCDIC translate table
- CPTEBCDIC | CEBCDIC uses an EBCDIC-only translate table
- CPTASCII | CASCII uses an ASCII-only translate table

Formatted data EZA trace records can be specified as follows:

EZADATA | EDATA | EZAEBCDIC | EZAASCII
or
EZADATA(*nn*) | EDATA(*nn*) | EZAEBCDIC(*nn*) | EZAASCII(*nn*)

- Where *nn* is between 0 and 65535 (default is 65535)

  Note: A data amount set very high generates a very large trace in a short period. A high value should be avoided unless you are absolutely convinced that there is a data integrity problem. For the vast majority of problems a glance value of 16 is sufficient. A value of 16 causes one line of data to displays.

- EZADATA | EDATA uses a mixed ASCII and EBCDIC translate table
- EZAEBCDIC | EEBCDIC uses an EBCDIC-only translate table
- EZAASCII | EASCII uses an ASCII-only translate table

FULL | SUMMARY

Show all or a summary of the data.

Default: SUMMARY.

GROUPS((*group* [, '*filter*']) ...)

Trace group or groups for which data is collected and optionally a filter parameter for each group.

Use the GROUPS parameter to limit the amount of data collected. Used with INSTANCE to modify an existing trace instance, either adds a new group to the trace instance or replaces an existing group for the trace instance. Once added, a group cannot be removed from the active trace instance.

For Unicenter Solve:CPT groups CPT and EZA are supported. Refer to the "Diagnostic Commands" chapter in the *Unicenter® TCPaccess™ Communications Server System Management Guide* for other TCPEEP group options that can be coded together with the Unicenter Solve:CPT options.

Note: You can specify a maximum of four trace groups.

group	Selects the type of data to collect: CPT or EZA **For TCPaccess only**: IUCV, NETIF.
GROUP(CPT,…	A collection of trace points in Unicenter Solve:CPT for processing CPT API activity with the following filter options: HOST(*host*,...,*host*)—Up to 16 IP HOST addresses (names). PORT(*port*,...,*port*)—Up to 16 port numbers (names). USER(*jobname*,..,*jobname*)—Takes 1 to 16 *jobnames* that refer to jobs using the CPT function. UASID(*asid*,...,*asid*)—Takes 1 to 16 *asids* that refer to jobs using the CPT function. TYPE([ACL | CLOSE] [, ACM | CONNLSTN] [, ADT | SENDRECV] [, AFM | GIVETAKE] [, AFT | CPTFTP] [, AXL | TRANSLAT] [, STR | STARTRAN] [, QUE | QUEUE] [MAXCPTDATA(*nnnn*) | MCPTDATA(*nnnn*) | MCDATA(*nnnn*)],)'

Where:

ACL|CLOSE is for CPT CLOSE calls.

ACM| CONNLSTN is for CONNECT and LISTEN calls.

ADT | SENDRECV is for SEND,RECEIVE, SENDTO and RCVFROM calls.

AFM | GIVETAKE is for GIVE and TAKE calls.

AFT | CPTFTP is for FTP calls.

AXL | TRANSLAT is for TRANSLATE calls.

[, STR | STARTRAN is for started transactions.

QUE | QUEUE is for READQ and WRiteq CALLS.

MAXCPTDATA(*nnnn*) | MCPTDATA(*nnnn*) | MCDATA(*nnnn*)-the *nnnn* value is a positive integer less than or equal to 65535.

GROUP(EZA,…

The EZA is a collection of trace points in Socket Management for processing EZASOKET API activity with the following filter options:

HOST(*host,...,host*)—Up to 16 IP HOST addresses (names).

PORT(*port,...,port*)—Up to 16 port numbers (names).

USER(*jobname,...,jobname*)—Takes 1 to 16 *jobnames* that refer to jobs using the EZA function.

UASID(*asid,...,asid*)—Takes 1 to 16 *asids* that refer to jobs using the EZA function.

MAXEZADATA(*nnnn*) | MEZADATA(*nnnn*) | MEDATA(*nnnn*)—Where *nnnn* is a positive integer less than or equal to 65535.

- A data amount set very high generates a very large trace in a short period. A high value should be avoided unless you are absolutely convinced that there is a data integrity problem. For the vast majority of problems a glance value of 16 is sufficient. A value of 16 causes one line of data to display.
- By default, no user data is traced. MAXEZADATA causes the data to be collected. It is better to limit EZA GROUP activity through the USER() and UASID() parameters rather than the PORT() and HOST() parameters. The PORT() and HOST() parameters require that the session be established or will loose all the API commands and output up to the point when a session is established.

HALT	Stops a component trace instance.
INSTANCE (*instance_id*)	Select a trace instance to display by specifying the *instance_id* returned from TCPEEP or the MVS TRACE CT command.
JOBNAME (*jobname* ,....)	Names of jobs used as filters for tracing. Events in these jobs are recorded by the component trace.
NOHEADER	Do not display the header information from the trace entry. This option helps limit the output. Without it, one line prints describing the trace entry even if no other information about that entry displays.
PEEK (*limit*)	Number of trace records to view. Use to view an existing trace. PEEK or PEEK(zero) implies no limit; PEEK(*n*) traces only *n* trace records. Range: Zero - no limit. Default: Zero.
SYSOUT (class)	Sends output to a dynamically allocated SYSOUT data set. By default, the output for TCPEEP writes to SYSTSPRT. Default: X.
TRACESIZE (*num_records*)	Optional. Maximum number of trace records to record. If not specified, there is no limit to the number of records recorded. TRACESIZE can only be specified when creating a trace instance. If specified when modifying a trace instance, it is ignored.
TRCSSID (*ssid*)	ID of the trace address space. Default: ACTR.

Suggested Tracing Invocation Options

Tracing should be turned on generally for events at the CPT, the API and the TCP level for most applications. This enables you to follow the data as it moves through different levels.

Suggested Unicenter SOLVE:CPT TCPEEP tracing invocation options when there is not a data integrity issue:

Note: Unicenter SOLVE:CPT tracing by default uses the ACL, ACM and ADT filter.

```
TCPEEP TRCSSID(ACTR)                                      +
FORMAT(TCP,DATA(32),IUCVDATA(32),EZADATA(32),CPTDATA(32)) +
GROUPS((CPT,’MAXCPTDATA(32)’),                            +
       (EZA,'MAXEZADATA(32)'),                            +
 (IUCV,'MAXIUCVDATA(32)'))
```

Note: You can use the IUCV keyword to specify an IUCV or HPNS application for tracing.

Since the defaults for TCPEEP tracing are ACL, ACM and ADT the previous TCPEEP should behave the same as if it were invoked using the following parameters:

```
TCPEEP TRCSSID(ACTR)                                      +
FORMAT(TCP,DATA(32),IUCVDATA(32),EZADATA(32),CPTDATA(32)) +
GROUPS((CPT,’TYPE(ACL,ACM,ADT),MAXCPTDATA(32)’),          +
       (EZA,'MAXEZADATA(32)'),                            +
 (IUCV,'MAXIUCVDATA(32)'))
```

However, should the TCPEEP be invoked with just a single parameter of the defaults then the other two trace type entries should not appear in the trace output. For example, the following TCPEEP trace call should contain ACM entries but not ACL or ADT entries:

```
TCPEEP TRCSSID(ACTR)                                      +
FORMAT(TCP,DATA(32),IUCVDATA(32),EZADATA(32),CPTDATA(32)) +
GROUPS((CPT,’TYPE(ACM),MAXCPTDATA(32)’),                  +
       (EZA,'MAXEZADATA(32)'),                            +
 (IUCV,'MAXIUCVDATA(32)'))
```

When there is a data integrity issue for a Unicenter SOLVE:CPT application then it is essential that you trace all the data as it moves through multiple layers. A sample TCPEEP for a data integrity issue would be of the format:

```
TCPEEP TRCSSID(ACTR)                                                  +
FORMAT(TCP,DATA(32000),IUCVDATA(32000),EZADATA(32000),CPTDATA(32000)) +
GROUPS((CPT,’TYPE(ACL,ACM,ADT,AXL,QUE),MAXCPTDATA(32000)’),           +
       (EZA,'MAXEZADATA(32000)'),                                     +
     (IUCV,'MAXIUCVDATA(32000)'))
```

You can turn on all Unicenter SOLVE:CPT tracing options by specifying:

```
TCPEEP TRCSSID(ACTR)                                                    +
FORMAT(TCP,DATA(32),IUCVDATA(32),EZADATA(32),CPTDATA(32))               +
GROUPS((CPT,’TYPE(ACL,ACM,ADT,AFM,AFT,AXL,STR,QUE),MAXCPTDATA(32)’),    +
       (EZA,'MAXEZADATA(32)'),                                          +
             (IUCV,'MAXIUCVDATA(32)'))
```

Where there are Unicenter SOLVE:CPT problems occurring while passing TCP sessions between multiple endpoints, use the following TCP options to debug the issue:

```
TCPEEP TRCSSID(ACTR)                                              +
FORMAT(TCP,DATA(32),IUCVDATA(32),EZADATA(32),CPTDATA(32))         +
GROUPS((CPT,’TYPE(ACL,ACM,ADT,AFM,STARTRAN),MAXCPTDATA(32)’),     +
       (EZA,'MAXEZADATA(32)'),                                    +
     (IUCV,'MAXIUCVDATA(32)'))
```

To debug Unicenter SOLVE:CPT FTP problems, use the following TCPEEP tracing options:

```
TCPEEP TRCSSID(ACTR)                                            +
FORMAT(TCP,DATA(32),IUCVDATA(32),EZADATA(32),CPTDATA(32))       +
GROUPS((CPT,’TYPE(ACL,ACM,ADT,STR,QUE,AFT),MAXCPTDATA(32)’),    +
       (EZA,'MAXEZADATA(32)'),                                  +
            (IUCV,'MAXIUCVDATA(32)'))
```

TCPEEP Example

The following example shows an excerpt of a trace that was generated by running the Unicenter SOLVE:CPT IVP(installation verification program). The TCPEEP parameters that were specified to generate this trace follow. Note they are the recommended trace parameters:

```
TCPEEP TRCSSID(ACTR)                                            +
FORMAT(TCP,DATA(32),IUCVDATA(32),EZADATA(32),CPTDATA(32))       +
GROUPS((CPT,’TYPE(ACL,ACM,ADT,STR,QUE,AFT),MAXCPTDATA(32)’),    +
       (EZA,'MAXEZADATA(32)'),                                  +
            (IUCV,'MAXIUCVDATA(32)'))
```

Sample TCPEEP output:

```
SOKENTER 000B0857 06/26 18:45:38.253616 EZA - EZASOKET Function Enter
080A QATS22R2 Entr Select                   Time In  06/26/2003 18:45:38.253587
  Conn: Prot(TCP)  Local=141.202.36.71:3024  Remote=0.0.0.0:0
  EZASoket 15E31C60 15E31410 15E31D20 15E31518
  EZASoket 15E31618 15E31718 15E31818 15E31918
  EZASoket 15E31A18 15E31408 95E3140C
  MaxSoc=50 Timeout: Sec(30)
  RSndMsk 00000001 00000000
  Tran=IPCP  Task=49  S(0)  UserID=NMDCICS   OrigToken=00000014
  CurrToken=00000014  CEP=15E33008  HIW=15E2EBC8  TCPIP=TCPIP71
   Subtask=F0F0F0F0F0F4F9C3
```

```
-----------------------------------------------------------
CPTEVENT 000D0207 06/26 18:45:42.402162 CPT - Queue Event
080A QATS22R2 Entr Queue Event             18:45:42.402162
  Q_Name=IPSF Q_Type=TD Q_Oper=WriteQ Q_RecLen=200 EIBresp=0
  CurrTranID=IPCK  Task#=64  SocDesc(0) UserID=HUSJOA2  Terminal=ACCVLT03
  CurrToken=00000000  TCPIPjob=TCPIP71   OrigToken=00000000
  Data +0000 C3969497 A4A38599 40C1A2A2 96838981 *Computer@Associa*
       +0010 A385A26B 40C9D5C3 4B404040 40404040 *tesk@INCK@@@@@@@*
       +0020 40404040 40404040 C3D7E340 C1D7C940 *@@@@@@@@CPT@API@*
       +0030 C3C9C3E2 40404040 40404040 C9D5E2E3 *CICS@@@@@@@@INST*
       +0040 C1D3D3C1 E3D6D540 E5C5D9C9 C6C9C3C1 *ALLATON@VERIFICA*
       +0050 E3C9D6D5 40D7D9D6 C7D9C1D4 404DC9E5 *TION@PROGRAM@MIV*
       +0060 D75D4040 F040F140 F240F340 F440F540 *P?@@0@1@2@3@4@5@*
       +0070 F640F740 F840F940 C1C2C3C4 C5C6C7C8 *6@7@8@9@ABCDEFGH*
       +0080 C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7 *IJKLMNOPQRSTUVWX*
       +0090 E8E98182 83848586 87888991 92939495 *YZabcdefghijklmn*
       +00A0 96979899 A2A3A4A5 A6A7A8A9 606D4B6B *opqrstuvwxyz`mKk*
       +00B0 7A5EE061 6C5B7B7C 4D5D4C6E 6F4E7E5A *z.\al?{.M?LnoN~Z*
       +00C0 C5D5C440 E3C5E2E3                   *END@TEST        *
-----------------------------------------------------------
CPTENTER 000D0201 06/26 18:45:42.467838 CPT-ACM Connection Functon Enter
080A QATS22R2 Entr Connect                  Time In  06/26/2003 18:45:42.462196
  Conn: Prot(TCP)  Local=0.0.0.0:0  Remote=0.0.0.0:0
  ACM Addr=95DA06C0  CPToken=00000000
  ACM Opts=800106 O3INC OCLEN NODNR LTRAN
  ACM LocIPaddr=0.0.0.0:0 ACM RemIPaddr=0.0.0.0:1350 Timeout: Sec(1)
  ACMSndBufSiz=1024 ACMRcvBufSiz=1024
  ACMucntx=16A01738
  CurrTranID=IPTS  Task#=65  SocDesc(-1)
  CurrToken=00000000  CEP=15E3E628  TCPIPjob=TCPIP71
-----------------------------------------------------------
SOKENTER 000B0857 06/26 18:45:42.471420 EZA - EZASOKET Function Enter
080A QATS22R2 Entr InitAPI                  Time In  06/26/2003 18:45:42.471344
  Conn: Prot(TCP)  Local=0.0.0.0:0  Remote=0.0.0.0:0
  EZASoket 00143760 15DA13E2 15DA13C8 15DA13D8
  EZASoket 15DA13E4 15DA13E8 95DA13EC
  MaxSock=50 Ident=TCPIP71 QATS22R2  Subtask=F0F0F0F0F0F6F5C3
  Tran=IPTS  Task=65  S(-1)  UserID=NMDCICS   OrigToken=00000000
  CurrToken=00000027  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F6F5C3
-----------------------------------------------------------
SOK-EXIT 000B1857 06/26 18:45:42.471800 EZA - EZASOKET Function Complete
080A QATS22R2 Cmpl InitAPI                  Time Out 06/26/2003 18:45:42.471795
  Conn: Prot(TCP)  Local=0.0.0.0:0  Remote=0.0.0.0:0
  EZASoket 00143760 15DA13E2 15DA13C8 15DA13D8
  EZASoket 15DA13E4 15DA13E8 95DA13EC
  RetCode(0) MaxSock=50 Ident=TCPIP71 QATS22R2  Subtask=F0F0F0F0F0F6F5C3 MaxSNO=49
  Tran=IPTS  Task=65  S(-1)  UserID=NMDCICS   OrigToken=00000000
  CurrToken=00000027  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F6F5C3
-----------------------------------------------------------
SOKENTER 000B0857 06/26 18:45:42.471864 EZA - EZASOKET Function Enter
080A QATS22R2 Entr Socket                   Time In  06/26/2003 18:45:42.471853
  Conn: Prot(TCP)  Local=0.0.0.0:0  Remote=0.0.0.0:0
  EZASoket 00143790 001437A0 001437A4 001437A8
  EZASoket 15DA13E8 95DA13EC
  SocType(TCP) NS=-1
  Tran=IPTS  Task=65  S(-1)  UserID=NMDCICS   OrigToken=00000000
  CurrToken=00000027  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F6F5C3
-----------------------------------------------------------
SOK-EXIT 000B1857 06/26 18:45:42.472255 EZA - EZASOKET Function Complete
080A QATS22R2 Cmpl Socket                   Time Out 06/26/2003 18:45:42.472251
  Conn: Prot(TCP)  Local=0.0.0.0:0  Remote=0.0.0.0:0
  EZASoket 00143790 001437A0 001437A4 001437A8
  EZASoket 15DA13E8 95DA13EC
  RetCode(0) SocType(TCP) NS=-1
  Tran=IPTS  Task=65  S(0)  UserID=NMDCICS   OrigToken=00000028
  CurrToken=00000028  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F6F5C3
```

```
-----------------------------------------------------------
SOKENTER 000B0857 06/26 18:45:42.473825 EZA - EZASOKET Function Enter
080A QATS22R2 Entr GetSockOpt              Time In  06/26/2003 18:45:42.473802
  Conn: Prot(TCP)  Local=0.0.0.0:0  Remote=0.0.0.0:0
  EZASoket 00143750 15DA13E0 001437AC 15DA1428
  EZASoket 15DA1424 15DA13E8 95DA13EC
  S(0) OptName(SO_SNDBUF) OptVal=00000000 OptLen(4)
  Tran=IPTS  Task=65  S(0)  UserID=NMDCICS   OrigToken=00000028
  CurrToken=00000028  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F6F5C3
-----------------------------------------------------------
SOK-EXIT 000B1857 06/26 18:45:42.474667 EZA - EZASOKET Function Complete
080A QATS22R2 Cmpl GetSockOpt              Time Out 06/26/2003 18:45:42.474660
  Conn: Prot(TCP)  Local=0.0.0.0:0  Remote=0.0.0.0:0
  EZASoket 00143750 15DA13E0 001437AC 15DA1428
  EZASoket 15DA1424 15DA13E8 95DA13EC
  RetCode(0) S(0) OptName(SO_SNDBUF) OptVal=00004000 OptLen(4)
  Tran=IPTS  Task=65  S(0)  UserID=NMDCICS   OrigToken=00000028
  CurrToken=00000028  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F6F5C3
-----------------------------------------------------------
SOKENTER 000B0857 06/26 18:45:42.474740 EZA - EZASOKET Function Enter
080A QATS22R2 Entr GetSockOpt              Time In  06/26/2003 18:45:42.474727
  Conn: Prot(TCP)  Local=0.0.0.0:0  Remote=0.0.0.0:0
  EZASoket 00143750 15DA13E0 001437B0 15DA1428
  EZASoket 15DA1424 15DA13E8 95DA13EC
  S(0) OptName(SO_RCVBUF) OptVal=00004000 OptLen(4)
  Tran=IPTS  Task=65  S(0)  UserID=NMDCICS   OrigToken=00000028
  CurrToken=00000028  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F6F5C3
-----------------------------------------------------------
SOK-EXIT 000B1857 06/26 18:45:42.474798 EZA - EZASOKET Function Complete
080A QATS22R2 Cmpl GetSockOpt              Time Out 06/26/2003 18:45:42.474794
  Conn: Prot(TCP)  Local=0.0.0.0:0  Remote=0.0.0.0:0
  EZASoket 00143750 15DA13E0 001437B0 15DA1428
  EZASoket 15DA1424 15DA13E8 95DA13EC
  RetCode(0) S(0) OptName(SO_RCVBUF) OptVal=00004000 OptLen(4)
  Tran=IPTS  Task=65  S(0)  UserID=NMDCICS   OrigToken=00000028
  CurrToken=00000028  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F6F5C3
-----------------------------------------------------------
SOKENTER 000B0857 06/26 18:45:42.474853 EZA - EZASOKET Function Enter
080A QATS22R2 Entr Bind                    Time In  06/26/2003 18:45:42.474837
  Conn: Prot(TCP)  Local=0.0.0.0:0  Remote=0.0.0.0:0
  EZASoket 00143710 15DA13E0 15DA1438 15DA13E8
  EZASoket 95DA13EC
  S(0) Name: Family(2) Port=0 IP=141.202.36.71
  Tran=IPTS  Task=65  S(0)  UserID=NMDCICS   OrigToken=00000028
  CurrToken=00000028  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F6F5C3
-----------------------------------------------------------
SOK-EXIT 000B1857 06/26 18:45:42.474983 EZA - EZASOKET Function Complete
080A QATS22R2 Cmpl Bind                    Time Out 06/26/2003 18:45:42.474979
  Conn: Prot(TCP)  Local=141.202.36.71:1116  Remote=0.0.0.0:0
  EZASoket 00143710 15DA13E0 15DA1438 15DA13E8
  EZASoket 95DA13EC
  RetCode(0) S(0) Name: Family(2) Port=0 IP=141.202.36.71
  Tran=IPTS  Task=65  S(0)  UserID=NMDCICS   OrigToken=00000028
  CurrToken=00000028  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F6F5C3
-----------------------------------------------------------
SOKENTER 000B0857 06/26 18:45:42.475037 EZA - EZASOKET Function Enter
080A QATS22R2 Entr Connect                 Time In  06/26/2003 18:45:42.475022
  Conn: Prot(TCP)  Local=141.202.36.71:1116  Remote=0.0.0.0:0
  EZASoket 00143720 15DA13E0 15DA1438 15DA13E8
  EZASoket 95DA13EC
  S(0) Name: Family(2) Port=1350 IP=127.0.0.1
  Tran=IPTS  Task=65  S(0)  UserID=NMDCICS   OrigToken=00000028
  CurrToken=00000028  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F6F5C3
```

```
-----------------------------------------------------------
SOK-EXIT 000B1857 06/26 18:45:42.482851 EZA - EZASOKET Function Complete
080A QATS22R2 Cmpl Connect                   Time Out 06/26/2003 18:45:42.482837
  Conn: Prot(TCP)  Local=141.202.36.71:1116  Remote=127.0.0.1:1350
  EZASoket 00143720 15DA13E0 15DA1438 15DA13E8
  EZASoket 95DA13EC
  RetCode(0) S(0) Name: Family(2) Port=1350 IP=127.0.0.1
  Tran=IPTS  Task=65  S(0)  UserID=NMDCICS   OrigToken=00000028
  CurrToken=00000028  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F6F5C3
-----------------------------------------------------------
SOKENTER 000B0857 06/26 18:45:42.488546 EZA - EZASOKET Function Enter
080A QATS22R2 Entr Select                    Time In  06/26/2003 18:45:42.488509
  Conn: Prot(TCP)  Local=141.202.36.71:1116  Remote=127.0.0.1:1350
  EZASoket 00143770 15E28F2C 15E28F30 15E28F98
  EZASoket 15E28FA0 15E28FA8 15E28FB0 15E28FB8
  EZASoket 15E28FC0 15DA13E8 95DA13EC
  MaxSoc=1 Timeout: Sec(1)
  WSndMsk 00000001
  Tran=IPTS  Task=65  S(0)  UserID=NMDCICS   OrigToken=00000028
  CurrToken=00000028  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F6F5C3
-----------------------------------------------------------
SOK-EXIT 000B1857 06/26 18:45:42.488724 EZA - EZASOKET Function Complete
080A QATS22R2 Cmpl Select                    Time Out 06/26/2003 18:45:42.488720
  Conn: Prot(TCP)  Local=141.202.36.71:1116  Remote=127.0.0.1:1350
  EZASoket 00143770 15E28F2C 15E28F30 15E28F98
  EZASoket 15E28FA0 15E28FA8 15E28FB0 15E28FB8
  EZASoket 15E28FC0 15DA13E8 95DA13EC
  RetCode(1) MaxSoc=1 Timeout: Sec(1)
  WRetMsk 00000001
  Tran=IPTS  Task=65  S(0)  UserID=NMDCICS   OrigToken=00000028
  CurrToken=00000028  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F6F5C3
-----------------------------------------------------------
CPTEXIT  000D1201 06/26 18:45:42.500596 CPT-ACM Connection Function Exit
080A QATS22R2 Cmpl Connect                   Time Out 06/26/2003 18:45:42.500579
  Conn: Prot(TCP)  Local=141.202.36.71:1116  Remote=127.0.0.1:1350
  ACM Addr=95DA06C0  CPToken=00000028  CPT RC(0) DiagCode(0)
  ACM Opts=800106 O3INC OCLEN NODNR LTRAN
  ACM LocIPaddr=141.202.36.71:1116 ACM RemIPaddr=127.0.0.1:1350 Timeout: Sec(1)
  ACMSndBufSiz=1024 ACMRcvBufSiz=1024
  ACMucntx=16A01738
  CurrTranID=IPTS  Task#=65  SocDesc(0) UserID=NMDCICS
  CurrToken=00000028  CEP=15E3E628  TCPIPjob=TCPIP71   OrigToken=00000028
-----------------------------------------------------------
CPTEVENT 000D0207 06/26 18:45:42.501512 CPT - Queue Event
080A QATS22R2 Entr Queue Event              18:45:42.501512
  Conn: Prot(TCP)  Local=141.202.36.71:1116  Remote=127.0.0.1:1350
  Q_Name=IPSF Q_Type=TD Q_Oper=ReadQ Q_RecLen=200 EIBresp=0
  CurrTranID=IPTS  Task#=65  SocDesc(0) UserID=NMDCICS
  CurrToken=00000028  CEP=15E3E628  TCPIPjob=TCPIP71   OrigToken=00000028
  Data +0000 C3969497 A4A38599 40C1A2A2 96838981 *Computer@Associa*
       +0010 A385A26B 40C9D5C3 4B404040 40404040 *tesk@INCK@@@@@@@*
       +0020 40404040 40404040 C3D7E340 C1D7C940 *@@@@@@@@CPT@API@*
       +0030 C3C9C3E2 40404040 40404040 C9D5E2E3 *CICS@@@@@@@@INST*
       +0040 C1D3D3C1 E3D6D540 E5C5D9C9 C6C9C3C1 *ALLATON@VERIFICA*
       +0050 E3C9D6D5 40D7D9D6 C7D9C1D4 404DC9E5 *TION@PROGRAM@MIV*
       +0060 D75D4040 F040F140 F240F340 F440F540 *P?@@0@1@2@3@4@5@*
       +0070 F640F740 F840F940 C1C2C3C4 C5C6C7C8 *6@7@8@9@ABCDEFGH*
       +0080 C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7 *IJKLMNOPQRSTUVWX*
       +0090 E8E98182 83848586 87888991 92939495 *YZabcdefghijklmn*
       +00A0 96979899 A2A3A4A5 A6A7A8A9 606D4B6B *opqrstuvwxyz`mKk*
       +00B0 7A5EE061 6C5B7B7C 4D5D4C6E 6F4E7E5A *z.\al?{.M?LnoN~Z*
       +00C0 C5D5C440 E3C5E2E3                   *END@TEST        *
```

```
-----------------------------------------------------------
CPTENTER 000D0202 06/26 18:45:42.502874 CPT - ADT Data Function Enter
080A QATS22R2 Entr Send                      Time In  06/26/2003 18:45:42.502839
  Conn: Prot(TCP)  Local=141.202.36.71:1116  Remote=127.0.0.1:1350
  ADT Addr=95DA0940  CPToken=00000028
  ADT Opts=00000000
  ADTbufAddr=15DA0DC8  ADTbuflen=200
  CurrTranID=IPTS  Task#=65  SocDesc(0) UserID=NMDCICS
  CurrToken=00000028  CEP=15E3E628  TCPIPjob=TCPIP71   OrigToken=00000028
  Data +0000 C3969497 A4A38599 40C1A2A2 96838981 *Computer@Associa*
       +0010 A385A26B 40C9D5C3 4B404040 40404040 *tesk@INCK@@@@@@@*
       +0020 40404040 40404040 C3D7E340 C1D7C940 *@@@@@@@@CPT@API@*
       +0030 C3C9C3E2 40404040 40404040 C9D5E2E3 *CICS@@@@@@@@INST*
       +0040 C1D3D3C1 E3D6D540 E5C5D9C9 C6C9C3C1 *ALLATON@VERIFICA*
       +0050 E3C9D6D5 40D7D9D6 C7D9C1D4 404DC9E5 *TION@PROGRAM@MIV*
       +0060 D75D4040 F040F140 F240F340 F440F540 *P?@@0@1@2@3@4@5@*
       +0070 F640F740 F840F940 C1C2C3C4 C5C6C7C8 *6@7@8@9@ABCDEFGH*
       +0080 C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7 *IJKLMNOPQRSTUVWX*
       +0090 E8E98182 83848586 87888991 92939495 *YZabcdefghijklmn*
       +00A0 96979899 A2A3A4A5 A6A7A8A9 606D4B6B *opqrstuvwxyz`mKk*
       +00B0 7A5EE061 6C5B7B7C 4D5D4C6E 6F4E7E5A *z.\al?{.M?LnoN~Z*
       +00C0 C5D5C440 E3C5E2E3                   *END@TEST        *
-----------------------------------------------------------
SOKENTER 000B0857 06/26 18:45:42.502962 EZA - EZASOKET Function Enter
080A QATS22R2 Entr Writev                    Time In  06/26/2003 18:45:42.502920
  Conn: Prot(TCP)  Local=141.202.36.71:1116  Remote=127.0.0.1:1350
  EZASoket 0015A0E0 15DA1238 15DA138C 15DA1388
  EZASoket 15DA13A4 95DA13A8
  S(0) IOVCnt=1
IOV 15DA0DC8 00000000 000000C8
 Tran=IPTS  Task=65  S(0)  UserID=NMDCICS   OrigToken=00000028
 CurrToken=00000028  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F6F5C3
 Data +0000 C3969497 A4A38599 40C1A2A2 96838981 *Computer@Associa*
      +0010 A385A26B 40C9D5C3 4B404040 40404040 *tesk@INCK@@@@@@@*
      +0020 40404040 40404040 C3D7E340 C1D7C940 *@@@@@@@@CPT@API@*
      +0030 C3C9C3E2 40404040 40404040 C9D5E2E3 *CICS@@@@@@@@INST*
      +0040 C1D3D3C1 E3D6D540 E5C5D9C9 C6C9C3C1 *ALLATON@VERIFICA*
      +0050 E3C9D6D5 40D7D9D6 C7D9C1D4 404DC9E5 *TION@PROGRAM@MIV*
      +0060 D75D4040 F040F140 F240F340 F440F540 *P?@@0@1@2@3@4@5@*
      +0070 F640F740 F840F940 C1C2C3C4 C5C6C7C8 *6@7@8@9@ABCDEFGH*
      +0080 C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7 *IJKLMNOPQRSTUVWX*
      +0090 E8E98182 83848586 87888991 92939495 *YZabcdefghijklmn*
      +00A0 96979899 A2A3A4A5 A6A7A8A9 606D4B6B *opqrstuvwxyz`mKk*
      +00B0 7A5EE061 6C5B7B7C 4D5D4C6E 6F4E7E5A *z.\al?{.M?LnoN~Z*
      +00C0 C5D5C440 E3C5E2E3                   *END@TEST        *
----------------------------------------------------------
OK-EXIT 000B1857 06/26 18:45:42.503351 EZA - EZASOKET Function Complete
80A QATS22R2 Cmpl Writev                     Time Out 06/26/2003 18:45:42.503346
 Conn: Prot(TCP)  Local=141.202.36.71:1116  Remote=127.0.0.1:1350
 EZASoket 0015A0E0 15DA1238 15DA138C 15DA1388
 EZASoket 15DA13A4 95DA13A8
 RetCode(200) S(0) IOVCnt=1
  IOV 15DA0DC8 00000000 000000C8
  Tran=IPTS  Task=65  S(0)  UserID=NMDCICS   OrigToken=00000028
  CurrToken=00000028  CEP=15E3E628  HIW=15E3FDF8  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F6F5C3
```

```
-----------------------------------------------------------
CPTEXIT  000D1202 06/26 18:45:42.503417 CPT - ADT Data Function Exit
080A QATS22R2 Cmpl Send                     Time Out 06/26/2003 18:45:42.503384
  Conn: Prot(TCP)  Local=141.202.36.71:1116  Remote=127.0.0.1:1350
  ADT Addr=95DA0940  CPToken=00000028  CPT RC(0) DiagCode(0)
  ADT Opts=00000000
  ADTbufAddr=15DA0DC8  ADTbuflen=200
  CurrTranID=IPTS  Task#=65  SocDesc(0) UserID=NMDCICS
  CurrToken=00000028  CEP=15E3E628  TCPIPjob=TCPIP71   OrigToken=00000028
  Data +0000 C3969497 A4A38599 40C1A2A2 96838981 *Computer@Associa*
       +0010 A385A26B 40C9D5C3 4B404040 40404040 *tesk@INCK@@@@@@@*
       +0020 40404040 40404040 C3D7E340 C1D7C940 *@@@@@@@@CPT@API@*
       +0030 C3C9C3E2 40404040 40404040 C9D5E2E3 *CICS@@@@@@@@INST*
       +0040 C1D3D3C1 E3D6D540 E5C5D9C9 C6C9C3C1 *ALLATON@VERIFICA*
       +0050 E3C9D6D5 40D7D9D6 C7D9C1D4 404DC9E5 *TION@PROGRAM@MIV*
       +0060 D75D4040 F040F140 F240F340 F440F540 *P?@@0@1@2@3@4@5@*
       +0070 F640F740 F840F940 C1C2C3C4 C5C6C7C8 *6@7@8@9@ABCDEFGH*
       +0080 C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7 *IJKLMNOPQRSTUVWX*
       +0090 E8E98182 83848586 87888991 92939495 *YZabcdefghijklmn*
       +00A0 96979899 A2A3A4A5 A6A7A8A9 606D4B6B *opqrstuvwxyz`mKk*
       +00B0 7A5EE061 6C5B7B7C 4D5D4C6E 6F4E7E5A *z.\al?{.M?LnoN~Z*
       +00C0 C5D5C440 E3C5E2E3                   *END@TEST        *
-----------------------------------------------------------
CPTENTER 000D0200 06/26 18:45:42.507194 CPT - ACL CLOSE Function Enter
080A QATS22R2 Entr Close                    Time In  06/26/2003 18:45:42.503459
  Conn: Prot(TCP)  Local=141.202.36.71:1116  Remote=127.0.0.1:1350
  ACL Addr=95DA0BC8  CPToken=00000028
  ACL Opts=00008000 C2INC
  CurrTranID=IPTS  Task#=65  SocDesc(0) UserID=NMDCICS
```

EZA TCPEEP Example

This section shows an excerpt of the output from a TCPEEP. This trace shows a small portion of a connection sequence from a loopback client into a CICS server. This is **not** a complete trace and was edited so that it could be included here. You will get similar results but not the exact entries when you run a trace.

Note: This is only meant as an example and is not all-inclusive. This includes the following sequence of trace events:

- Server finishing issuing a listen call
- Server issuing a select call waiting for connections
- Client issuing an initapi call
- Client issuing a socket call
- Skip of Client issuing get type calls
- Client issuing a connect call

Note: The trace parameters are displayed at the beginning of the trace.

```
  READY
  TCPEEP TRCSSID(ACQA) FORMAT(TCP,DATA(16),IUCVDATA(16),EZADATA(16))
GROUPS((EZA,'MAXEZADATA(16)')         (IUCV,'MAXIUCVDATA(16)')
 )
 T03PE010I TRACE INITIATED - INSTANCE:  01
-----------------------------------------------------------
 SOK-EXIT 000B1857 04/02 12:52:55.737764 EZA - EZASOKET Function Complete
 0823 LONCICS5 Cmpl LISTEN                   TIME OUT 04/02/2002 12:52:55.737753
   Conn: Prot(TCP)  Local=0.0.0.0:1234  Remote=0.0.0.0:0
   EZASOKET 151E6B65 1540091E 1540092C 15400924
   EZASOKET 95400928
   RETCODE(0) S(0) BACKLOG=5
   TRAN=IPTL  TASK=36  S(0)  USERID=MEL2
   TOKEN=00000005  CEP=153E8008  HIW=153DFC88  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F3F6C3
 -----------------------------------------------------------
 SOKENTER 000B0857 04/02 12:52:55.737818 EZA - EZASOKET Function Enter
 0823 LONCICS5 Entr SELECT                  12:52:55.737818
   Conn: Prot(TCP)
   EZASOKET 151E6B85 15400930 151E6BD4 154009B8
   EZASOKET 15400AB8 15400BB8 15400CB8 15400DB8
   EZASOKET 15400EB8 15400924 95400928
   MAXSOC=50 TIMEOUT: SEC(-1) USEC(-1)
   RSNDMSK 00000001 00000000 000A
-----------------------------------------------------------
 SOKENTER 000B0857 04/02 12:54:13.593463 EZA - EZASOKET Function Enter
 0823 LONCICS5 Entr INITAPI                  TIME IN  04/02/2002 12:54:13.593357
   EZASOKET 00197B97 15418034 1541805C 15418038
   EZASOKET 15418070 15418024 95418020
   MAXSOCK=256 IDENT=                 Subtask=F0F0F0F0F0F3F8E2
   TRAN=JSC1  TASK=38  S(-1)  USERID=MEL2     TERMINAL=A03VLT36
   TOKEN=00000006  CEP=153EB008  HIW=153E8D68  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F3F8E2
 -----------------------------------------------------------
 SOK-EXIT 000B1857 04/02 12:54:13.826950 EZA - EZASOKET Function Complete
 0823 LONCICS5 Cmpl INITAPI                  TIME OUT 04/02/2002 12:54:13.826919
   EZASOKET 00197B97 15418034 1541805C 15418038
   EZASOKET 15418070 15418024 95418020
   RETCODE(0) MAXSOCK=256 IDENT=                 Subtask=F0F0F0F0F0F3F8E2 MAXSNO=255
   TRAN=JSC1  TASK=38  S(-1)  USERID=MEL2     TERMINAL=A03VLT36
   TOKEN=00000006  CEP=153EB008  HIW=153E8D68  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F3F8E2
 -----------------------------------------------------------
 SOKENTER 000B0857 04/02 12:54:13.827098 EZA - EZASOKET Function Enter
 0823 LONCICS5 Entr SOCKET                   TIME IN  04/02/2002 12:54:13.827063
   EZASOKET 00197A90 15417E90 15417E94 15417E98
   EZASOKET 15417E64 95417E60
   SOCTYPE(TCP) NS=-1
   TRAN=JSC1  TASK=38  S(-1)  USERID=MEL2     TERMINAL=A03VLT36
   TOKEN=00000006  CEP=153EB008  HIW=153E8D68  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F3F8E2
 -----------------------------------------------------------
 SOK-EXIT 000B1857 04/02 12:54:13.829499 EZA - EZASOKET Function Complete
 0823 LONCICS5 Cmpl SOCKET                   TIME OUT 04/02/2002 12:54:13.829481
   Conn: Prot(TCP)  Local=0.0.0.0:0  Remote=0.0.0.0:0
   EZASOKET 00197A90 15417E90 15417E94 15417E98
   EZASOKET 15417E64 95417E60
   RETCODE(0) SOCTYPE(TCP) NS=-1
   TRAN=JSC1  TASK=38  S(0)  USERID=MEL2     TERMINAL=A03VLT36
   TOKEN=00000007  CEP=153EB008  HIW=153E8D68  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F3F8E2
 -----------------------------------------------------------
Skipped>>>>>0823 LONCICS5 Entr GETHOSTBYNAME  12:54:13.829660
Skipped>>>>>0823 LONCICS5 Entr GETHOSTID   TIME IN  04/02/2002 12:54:14.044875
Skipped>>>>>0823 LONCICS5 Entr GETHOSTBYADDR TIME IN  04/02/2002 12:54:14.045520
-----------------------------------------------------------
 SOKENTER 000B0857 04/02 12:54:14.069114 EZA - EZASOKET Function Enter
 0823 LONCICS5 Entr BIND                      TIME IN  04/02/2002 12:54:14.069047
   Conn: Prot(TCP)  Local=141.202.36.71:0  Remote=0.0.0.0:0
   EZASOKET 00197726 154121B2 15417E98 15417E6C
   EZASOKET 95417E68
   S(0) NAME: Family(2) Port=0 IP=0.0.0.0
   TRAN=JSC1  TASK=38  S(0)  USERID=MEL2     TERMINAL=A03VLT36
   TOKEN=00000007  CEP=153EB008  HIW=153E8D68  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F3F8E2
 -----------------------------------------------------------
 SOK-EXIT 000B1857 04/02 12:54:14.069272 EZA - EZASOKET Function Complete
 0823 LONCICS5 Cmpl BIND                      TIME OUT 04/02/2002 12:54:14.069262
   Conn: Prot(TCP)  Local=0.0.0.0:3055  Remote=0.0.0.0:0
   EZASOKET 00197726 154121B2 15417E98 15417E6C
   EZASOKET 95417E68
   RETCODE(0) S(0) NAME: Family(2) Port=0 IP=0.0.0.0
   TRAN=JSC1  TASK=38  S(0)  USERID=MEL2     TERMINAL=A03VLT36
   TOKEN=00000007  CEP=153EB008  HIW=153E8D68  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F3F8E2
 -----------------------------------------------------------
 SOKENTER 000B0857 04/02 12:54:14.069364 EZA - EZASOKET Function Enter
 0823 LONCICS5 Entr CONNECT                   TIME IN  04/02/2002 12:54:14.069345
   Conn: Prot(TCP)  Local=0.0.0.0:3055  Remote=0.0.0.0:0
   EZASOKET 00197756 154121B2 15417E98 15417E6C
   EZASOKET 95417E68
   S(0) NAME: Family(2) Port=1234 IP=141.202.36.71
   TRAN=JSC1  TASK=38  S(0)  USERID=MEL2     TERMINAL=A03VLT36
   TOKEN=00000007  CEP=153EB008  HIW=153E8D68  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F3F8E2
 -----------------------------------------------------------
 SOK-EXIT 000B1857 04/02 12:54:14.069614 EZA - EZASOKET Function Complete
 0823 LONCICS5 Cmpl CONNECT                   TIME OUT 04/02/2002 12:54:14.069602
   Conn: Prot(TCP)  Local=0.0.0.0:3055  Remote=141.202.36.71:1234
   EZASOKET 00197756 154121B2 15417E98 15417E6C
   EZASOKET 95417E68
   RETCODE(0) S(0) NAME: Family(2) Port=1234 IP=141.202.36.71
   TRAN=JSC1  TASK=38  S(0)  USERID=MEL2     TERMINAL=A03VLT36
   TOKEN=00000007  CEP=153EB008  HIW=153E8D68  TCPIP=TCPIP71   Subtask=F0F0F0F0F0F3F8E2
```

TRACE

The IBM MVS Component Trace facility was enhanced to provide a method of collecting TCP/IP data and displaying it on a terminal or sending it to an external writer. Additional JCL is required to enable the Component Trace enhancements. To use Component Trace as a TSO command, see TCPEEP.

MVS Component Trace

MVS Component Trace is a diagnostic aid used to trace the action of certain system components and third party components that define themselves to Component Trace. The TRACE operator command is used to start, stop and control the component trace. For more information on the MVS TRACE command, see the IBM MVS System Commands.

Trace Address Space

Component Trace is defined in its own address space and collects trace data for trace points defined in other address spaces. Likewise, the other address spaces must identify the Component Trace address space for the data collection. Multiple occurrences of Component Trace can be active at the same time, each with a unique subsystem ID.

If you are running Unicenter TCPaccess, for each address space, IJTCFG*xx* in the PARM member defines the subsystem ID. The definition is specified using the TRACENAME keyword on the IFSPARM statement as follows.

In the following example, the trace data in the address space is directed to the Component Trace address space with an ACTR subsystem ID.

```
IFSPARM PROMPT VMCFNAME( VMCF ) TRACENAME( ACTR ) NOPROMPT
```

Note: If the Trace Address Space is brought up after the TCP stack, there may be a delay of two minutes before events are traced.

Exit

MVS Component Trace requires an exit to communicate with the tracing component. The T03PTRSS exit must reside in LPALIB or the linklist. It is distributed in the LINK library.

External Writer

The collected trace data is written to DASD or TAPE using an external writer. A suitable External Writer Cataloged Procedure for use with Component Trace may already be defined on your system. For more information on defining an External Writer used with Component Trace, see the IBM publication *MVS Diagnosis: Tools and Service Aids*.

Component TraceJCL

The JCL to run a Component Trace Address Space can be copied from the T09SAMP library member T09TRACE. The following is sample JCL for running trace.

CAUTION! *Be sure to specify P=T03 (shown in bold below) on the TRACE statement to specify the Component Trace Address space. If you specify T01, you will bring up a second Unicenter TCPaccess stack.*

```
//T09TRACE JOB
//*
//*  SAMPLE JCL PROCEDURE TO RUN SOCKET MANAGEMENT TRACE ADDRESS SPACE
//*  NOTE:  THIS ADDRESS SPACE SHOULD NOT BE TERMINATED
//*
//*  EDIT THE TRGINDX, SSN, SOUT, CMND SYMBOLIC PARAMETERS
//*
//*  VERIFY THAT THE JOB CARD AND NAMING CONVENTIONS MEET
//*  YOUR SITE'S JCL REQUIREMENTS, THEN SUBMIT THIS JOB.
//*
//ICSTRACE PROC TRGINDX='TRGINDX', TARGET LIBRARIES DSN INDEX
//            SSN=ACTR,            DFLT SUBSYSTEM NAME
//            SOUT='*',            CHOOSE A HOLD NONPURGE SYSOUT CLASS
//            CMND=STARTTR,        DFLT STARTUP COMMAND SCRIPT NAME
//            CNFG=00              IJTCFGXX SUFFIX
//*
//TRACE    EXEC PGM=IFSSTART,REGION=6144K,TIME=1440,
// PARM='IFSINIT,U=&SSN,P=T03,SO=&SOUT,CM=&CMND,CF=&CNFG'
//*
//STEPLIB  DD  DISP=SHR,DSN=&TRGINDX..LOAD
//*
//* WARNING: THE LOAD DATA SET MUST NEVER BE ADDED TO THE LINK LIST.
//*          COMPONENT TRACE ELEMENT NAMES ARE NOT UNIQUE AND MAY
//*          AFFECT THE OPERATIONS OF OTHER SOFTWARE.
//*          THE LIBRARY SHOULD ALWAYS BE REFERENCED THROUGH
//*          A STEPLIB OR JOBLIB STATEMENT.
//*
//*        CONFIGURATION DATA SETS
//*
//SYSPARM  DD  DISP=SHR,DSN=&TRGINDX..PARM
//SYSPROC  DD  DISP=SHR,DSN=&TRGINDX..PARM
//SYSHELP  DD  DISP=SHR,DSN=&TRGINDX..HELP
//*
//*        LOG DATA SETS
//*
//T01LOG   DD  SYSOUT=&SOUT
//SYSPRINT DD  SYSOUT=&SOUT
//*
//*        DUMP DATA SETS
//*
//SYSUDUMP DD  SYSOUT=&SOUT
//*
//*        MISC DATA SETS
//*
//ABNLIGNR DD  DUMMY         /* DISABLE ABEND-AID PROCESSING */
//*
```

TRACE Command

The TRACE command starts, stops, modifies, or displays the status of a system trace, master trace, or component trace.

The TRACE command is a standard MVS operator command.

```
TRACE    [ CT { [ , ON | OFF ] [ , COMP=name ] [ , PARM=mem ] } ]
         [ CT { [ , WTRSTART=mem_name[,WRAP | NOWRAP ] ] } ]
         [ CT { , WTRSTOP=job_name } ] ]
```

ON

Turns on tracing for a component if the component trace is currently off. If the component trace is on and can be changed, this changes the trace options.

OFF

Turns off tracing for the component. If the component is connected to an external writer, the trace is implicitly disconnected from the writer.

COMP=*name*

Identifies the component trace with the subsystem ID for the trace address space. This is required for each TRACE command.

PARM=*mem*

Identifies a member of SYS1.PARM or a data set in the system parmlib concatenation containing the parameters used for tracing. Using a parmlib member enables the operator to initiate the trace, change it, or stop it without a message prompting for parameters.

Parameters specified on the TRACE command override the options specified in the parmlib member. The parameters are described in TRACE Command Reply .

WTRSTART= *mem_name*

Identifies the member containing the JCL to invoke an external writer and opens the data sets used by the external writer. The member must be a SYS1.PROCLIB cataloged procedure or a job.

After starting the external writer, use the WTR parameter to connect the component trace to the external writer.

WRAP | NOWRAP

NOWRAP instructs the system to stop writing data to a data set when the data set is full. With the WRAP parameter, when the data set or group of data sets is full, new data overwrites the oldest data at the start of the data set or the start of the first data set.

If the WTRSTART parameter on the TRACE CT command specifies NOWRAP, the system uses the primary and secondary extents of the data set or sets. If the WTRSTART parameter specifies WRAP or omits the parameter, the system uses only the primary extent or extents.

WTRSTOP=*job_name* Disconnects the external writer from the component trace and closes the data sets used by the external writer.

jobname is the member name if the source JCL is a procedure, or a job name if defined on a JOB statement within the source JCL.

Before stopping the external writer, turn the component trace off with **TRACE CT,OFF** or disconnect the external writer with **WTR=DISCONNECT**.

TRACE Command Reply

In response to a TRACE CT,ON command without the PARM parameter, the system prompts you to specify the component trace options. Use the REPLY command to respond.

```
R id[ ,ASID=( nnnn [ ,nnnn ]...) ]
        [ ,CONT | ,END ]
        [,JOBNAME=( name [ ,name ]...) ]
        [ ,OPTIONS=( option [ ,option ]...) ]
        [ ,WTR={ mem_name | DISCONNECT } ]
```

id Use the same identification number (0-9999) from the message to identify the reply.

ASID**=**(*nnnn* **[,** *nnnn* **] ...**)

Specifies the address space identifiers, ASIDs, of address spaces used as a filter for tracing. Events in the ASIDs are recorded by the component trace.

The parameter contains a list of 0 to 16 hexadecimal ASIDs separated by commas. An empty ASID list, ASID=(), turns off filtering by address spaces.

In the ASID parameter, list all address spaces to be traced. Address spaces for previous traces are not traced unless listed.

CONT or END

CONT	Continues the reply on another line. The system issues another reply message. You can then continue the reply and repeat any parameters on the continuation line, except END. Repeated parameters are strung together; they do not overlay each other.
END	Identifies the end of the REPLY.

Note: CONT or END must be the last parameter on the input line.

JOBNAME=(*name* [**,** *name*] **...**)

Names of jobs used as filters for tracing. Events in these jobs are recorded by the component trace.

The parameter contains a list of 0 to 16 job names separated by commas.

An empty job list, JOBNAME=(), turns off filtering by jobs.

In the JOBNAME parameter, list all jobs to be traced. Jobs specified for previous traces are not traced unless listed.

OPTIONS=(*option* [, *option*] **...**)

Specifies the component trace options described in TRACE Command Reply Options.

WTR= *mem_name* | DISCONNECT

mem_name	Identifies the member containing the source JCL that invokes the external writer. The member must be a SYS1.PROCLIB cataloged procedure or a job. The *mem_name* in the WTR parameter must match the *mem_name* in the TRACE CT, WTRSTART command.
WTR=DISCONNECT	Disconnects the external writer. The component continues tracing and placing the trace records in the address-space buffer, but stops passing trace records to the external writer.

Note: You must also specify a TRACE CT,WTRSTART or TRACE CT,WTRSTOP command to start or stop the writer.

TRACE Command Reply Options

Use the Options parameter in response to the Reply prompt.

```
OPTIONS= [ BUFFERS( size,num ) ] [ BUFFTIME( time ) ] [ DATASIZE( dsize ) ]
         [ GROUPS( ( group[, 'filter' ] )...) ] [ HALT ]
         [ INSTANCE( inst ) ] [ STATUS ] [ TRACESIZE( tsize ) ] [ WRAP | NOWRAP ]
```

BUFFERS (*size*,*num*)

Optional. Size of the trace buffers in kilobytes or the number of buffers.

size	A value between 64 and 1024. Default: 256.
num	A value between 2 and 128. Default: Four.

BUFFERS can only be specified for a new trace instance.

Note: If specified when modifying a trace instance, it is ignored.

BUFFTIME(*time*)

Optional. Buffer timeout interval in seconds. At the end of each interval, if the current buffer contains data but is not full, a buffer flush operation is initiated.

BUFFTIME is considered only when creating a trace instance.

Use this parameter to force a buffer switch so you do not have to wait for the entire buffer to fill to see trace data.

Range: 0 – 99999.

Default: 10.

DATASIZE(*dsize*)

Optional. Specifies the maximum size of a trace record in kilobytes. Trace records that exceed the specified value are truncated.

DATASIZE is optional and can only be specified when creating a trace instance. If specified when modifying a trace instance, it is ignored. If the specified maximum size exceeds the largest supported trace record size (64 KB less control headers), then the specification has no effect.

Default: Any size record is recorded (up to the maximum IBM limit of 64 KB less control headers.)

GROUPS((*group* [, '*filter*']) ...)

Trace group or groups for which data is collected and optionally a filter parameter for each group.

Use the GROUPS parameter to limit the amount of data collected. Used with INSTANCE to modify an existing trace instance, either adds a new group to the trace instance or replaces an existing group for the trace instance. Once added, a group cannot be removed from the active trace instance.

For Unicenter NetMaster Socket Management for CICS, only group EZA is supported. Refer to the “Diagnostic Commands” chapter in the *Unicenter® TCPaccess™ Communications Server System Management Guide* for other TCPEEP group options that can be coded with the Unicenter NetMaster Socket Management for CICS options.

Note: You can specify a maximum of four trace groups.

group	Selects the type of data to collect: CPT, EZA **For TCPaccess only:** IUCV, NETIF.

GROUP(CPT,…	A collection of trace points in Unicenter SOLVE:CPT for processing CPT API activity with the following filter options:

HOST(*host,...,host*)—Up to 16 IP HOST addresses (names).

PORT(*port,...,port*)—Up to 16 port numbers (names).

USER(*jobname,..,jobname*)—Takes 1 to 16 job names that refer to jobs using the EZA function.

UASID(*asid,...,asid*)—Takes 1 to 16 *asids* that refer to jobs using the EZA function.
TYPE(
[ACL | CLOSE]
[, ACM | CONNLSTN]
[, ADT | SENDRECV]
[, AFM | GIVETAKE]
[, AFT | CPTFTP]
[, AXL | TRANSLAT]
[, STR | STARTRAN]
[, QUE | QUEUE]
[MAXCPTDATA(*nnnn*) | MCPTDATA(*nnnn*) |
MCDATA(*nnnn*)],
)

where:

ACL|CLOSE is for CPT CLOSE calls.

ACM| CONNLSTN is for CONNECT and LISTEN calls.

ADT | SENDRECV is for SEND, RECEIVE, SENDTO and RCVFROM calls.

AFM | GIVETAKE is for GIVE and TAKE calls.

AFT | CPTFTP is for FTP calls.

AXL | TRANSLAT is for TRANSLATE calls.

[, STR | STARTRAN is for started transactions.

QUE | QUEUE is for READQ and WRiteq CALLS.

MAXCPTDATA(*nnnn*) | MCPTDATA(*nnnn*) | MCDATA(*nnnn*)—the *nnnn* value is a positive integer less than or equal to 65535.

GROUP(EZA,…

The EZA is a collection of trace points in Unicenter NetMaster Socket Management for CICS for processing EZASOKET API activity with the following filter options:

HOST(*host*,...,*host*)—Up to 16 IP HOST addresses (names).

PORT(*port*,...,*port*)—Up to 16 port numbers (names).

USER(*jobname*,..,*jobname*)—Takes 1 to 16 job names that refer to jobs using the EZA function.

UASID(*asid*,...,*asid*)—Takes 1 to 16 asids that refer to jobs using the EZA function.

MAXEZADATA*(nnnn)* | MEZADATA(*nnnn)* | MEDATA(*nnnn*)—Where nnnn is a positive integer less than or equal to 65535.

Notes:

- A data amount set very high generates a very large trace in a short period. A high value should be avoided unless you are absolutely convinced that there is a data integrity problem. For the vast majority of problems a glance value of 16 is sufficient. A value of 16 causes one line of data to display.
- By default, no user data is traced, so MAXEZADATA causes the data to be collected.

  It is better to limit EZA GROUP activity through the USER() and UASID() parameters rather than the PORT() and HOST() parameters. The PORT() and HOST() parameters require that session to be established or you will loose all the API commands and output up to the point when a session was established.

HALT

Specifies that the trace instance identified by the INSTANCE keyword should be stopped.

If HALT is specified, INSTANCE must be specified. If the INSTANCE is connected to an external writer, WTR=DISCONNECT must also be specified.

INSTANCE (*inst*)

Modifies or stops a trace instance.

inst

The component trace address space ID. A new trace instance is created if *inst* is not specified.

STATUS

Displays all active trace instances.

Note: If STATUS is specified, any other keywords are ignored.

TRACESIZE(*tsize*)	Optional. Maximum number of trace records to record. If not specified, there is no limit to the number of records recorded. TRACESIZE can only be specified when creating a trace instance. If specified when modifying a trace instance, it is ignored.
WRAP | NOWRAP	WRAP creates an in-memory WRAP trace. Not valid with WTR=*mem_name*, nor can the instance be modified later to specify WTR=*mem_name*. This is the default if NOWRAP, is not specified and WTR=*mem_name* is omitted. NOWRAP creates a NOWRAP trace. This is the default if WTR=*mem_name* is specified. The external writer writes the buffers when they fill up and after being written are reused. This option is for the Trace Address space and should not be confused with WRAP|NOWRAP on the TRACE CT command, which is for the IBM writer.

Comparing TCPEEP and MVS TRACE Syntax

The syntax for commands using TCPEEP differ from those used with the MVS TRACE facility.

OPTIONS

To use MVS TRACE to specify EZA and MEDATA(32), you need to specify:

```
OPTIONS = ( GROUPS ( ( EZA, ' MAXEZADATA ( 32 ) ' ) ) )
```

This is the same as the TSO TCPEEP command:

```
TCPEEP GROUPS ( ( EZA, ' MAXEZADATA ( 32 ) ' ) )
```

JOBNAME

To specify JOBNAME using the MVS TRACE command, the syntax is:

```
JOBNAME = ( jobname1, ..., jobname16 )
```

When using TCPEEP the syntax is

```
TCPEEP JOBNAME ( jobname1, ..., jobname16 )
```

For more information about TCPEEP, see TCPEEP.

TRACE Command Examples

The following examples demonstrate the control of the T03 Trace Facility using the MVS TRACE command. The examples assume a T03 Trace Facility is active and is using a Subsystem ID of ACTR.

Starting a Trace Instance

In this example a new trace instance is created. The instance is limited to tracing 10000 records for group ID EZA from address space CICSPROD.

```
TRACE CT,ON,COMP=ACTR
R xx,OPTIONS=( GROUPS(EZA )  TRACESIZE( 10000 ) ),JOBNAME=(CICSPROD),END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR909I Trace start successful Instance(01)
```

Modifying a Trace Instance

In this example an existing trace instance is modified. The trace instance is changed to include records for group ID EZA.

```
TRACE CT,ON,COMP=ACTR
R xx,OPTIONS=( INSTANCE( 1 ) GROUPS( EZA ) ),END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR911I Trace modify successful Instance(01)
```

Displaying Trace Status

In this example the status of the trace instances displays.

```
TRACE CT,ON,COMP=ACTR
R xx,OPTIONS=( STATUS ),END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR916I Instance(01) Active, records=23,745
T03TR916I Instance(02) Active, records=1,576
```

Stopping a Trace Instance

In this example an existing trace instance is stopped.

```
TRACE CT,ON,COMP=ACTR
R xx,OPTIONS=( INSTANCE( 1 ) HALT ),END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR910I Trace shutdown successful Instance(01)
```

Stopping All Trace Instances

In this example all existing trace instances are stopped.

```
TRACE CT,OFF,COMP=ACTR
T03TR910I Trace shutdown successful Instance(01)
T03TR910I Trace shutdown successful Instance(02)
```

Starting an External Writer

In this example an external writer is started. T03XWTR is the name of a predefined started task.

```
TRACE CT,WTRSTART=T03XWTR
```

Starting a Trace Instance and Connecting an External Writer

In this example, a new trace instance is created and connected to an external writer. Only one instance can have an external writer connected.

```
TRACE CT,ON,COMP=ACTR
R xx,OPTIONS=( GROUPS( NETIF ) ),WTR=T03XWTR,END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR909I Trace start successful Instance(01)
```

Modifying a Trace Instance to Connect an External Writer

In this example an existing trace instance is connected to an external writer. Only one instance can have an external writer connected.

```
TRACE CT,ON,COMP=ACTR
R xx,OPTIONS=( INSTANCE( 1 ) ),WTR=T03XWTR,END
```

Where *xx* ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND. The resulting message is:

```
T03TR911I Trace modify successful Instance(01)
```

Modifying a Trace Instance to Disconnect an External Writer

In this example an existing trace instance is disconnected from an external writer.

```
TRACE CT,ON,COMP=ACT
RR xx,OPTIONS=( INSTANCE( 1 ) ),WTR=DISCONNECT,END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR911I Trace modify successful Instance(01)
```

Stopping a Trace Instance and Disconnecting an External Writer

In this example an existing trace instance is to be stopped and disconnected from an external writer.

```
TRACE CT,ON,COMP=ACT
RR xx,OPTIONS=( INSTANCE( 1 ) HALT ),WTR=DISCONNECT,END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR910I Trace shutdown successful Instance(01)
```

Stopping an External Writer

In this example an external writer is stopped. The writer must be disconnected from a trace instance before it can be stopped.

```
TRACE CT,WTRSTOP=T03XWTR
```

Sample TRACE Instance for Debugging

In this section, we show an example of just the trace component perspective, for full instructions read and follow section: Diagnostic Procedures in the *Troubleshooting* chapter.

Issue the following MVS console operator command to turn on system CT tracing where the subsystem of the trace address space is ACTR:

```
/TRACE CT,ON,COMP=ACTR
```

At this time, the operator will be prompted by the trace address space to enter the appropriate tracing related variables. Enter the GROUPS and BUFFERS related parameters in the OPTIONS field.

The GROUPS parameters are the same as the GROUPS parameters in a TCPEEP command.

The BUFFERS(SIZE,NUMBER) variable in the trace address space allocates storage for the tracing. Setting BUFFERS to a large SIZE and NUMBER reserves more storage for tracing.

SIZE is in kilobytes and ranges from 64 to 1024. NUMBER ranges from 2 to 128.

In this example, we want TCPEEP to trace Unicenter SOLVE:CPT data for port 3666 and CPT, EZA and IUCV data for user ID CICSPROD and the first 16 bytes of data of TCP tracing:

```
R 82,OPTIONS=(GROUPS((NETIF,'PROTOCOL(TCP),PORT(3666),MDATA(16)'),
(CPT,'USER(CICSPROD),MCPTDATA(16)’),(EZA,'USER(CICSPROD),MEZADATA(16)'),
(IUCV,'USER(CICSPROD),MIDATA(16)')),BUFFERS(256,64)),END
```

At the point when the failure occurs, you want to take an SVC dump of the trace address space to gather the TCPEEP data before it internally wraps. Use the SVCDUMP command to dump a hang and a SLIP trap for an ABEND situation: for example, S0C4.

If the TCP/IP stack is Unicenter TCPaccess with a jobname of RUNTCP and the trace address space jobname is RUNTRACE then the following operator command enables you to dump the application address space named CICSPROD, the trace address space and Unicenter TCPaccess:

```
F RUNTCP,SVCDUMP JOBNAME(RUNTRACE,CICSPROD)
```

The following command under IPCS for the SVC dump output can format the data using the GROUP and FORMAT parameters (the same parameters as used with TCPEEP) are placed in the OPTIONS field where the subsystem of the trace address space is ACTR:

```
CTRACE SUMMARY COMP(ACTR) OPTIONS((FORMAT(TCP,DATA(16),IDATA(16),CDATA(16)),
GROUPS((NETIF,'PROTOCOL(TCP),MDATA(16)'),(IUCV,'MIDATA(16)'),
(CPT,’MCPTDATA(16)),(EZA,’MEZADATA(16)))))
```

Entering Long Trace Replies

There are cases in which you want to create a high level of filtering in order to limit the size of the trace data being collected. It is an MVS restriction to limit the reply to the trace prompt to 126 characters in total length. The best way around this restriction is to enter and modify the original trace instance.

The following set of operator commands is equivalent of entering the following trace options reply all at once if you were not limited to entering 126 characters on a command line:

```
/R 82,OPTIONS=(GROUPS((NETIF,'PROTOCOL(TCP),PORT(1350),MDATA(16)'),
(CPT,'USER(CICSPROD),TYPE(ACL,ACM,ADT,AFM,STR),MCPTDATA(16)’),(EZA,'USER(CICSPROD)
,MEZADATA(16)'),(IUCV,'USER(CICSPROD),MIDATA(16)')),BUFFERS(256,64)),END
```

In the example the following is assumed:

- The trace subsystem is ACTR.
- The TCPaccess job is called RUNTCP.
- The Trace address space job is called RUNTRACE.
- The CICS job call is CICSPROD.
- The only port we are interested in tracing is port 1350.
- The trace instance is 1 after the first trace operator command.
- The CPT trace entries we are interested in are for CLOSE (ACL), LISTEN (ACM), SEND and RECEIVE (ADT), GIVE and TAKE events (AFM) and started task events (STR).

1. Enter the following command and response to get the CT trace going for tracing subsytem ACTR for NETIF, IUCV, EZA and CPT tracing with a large buffer pool for tracing

   ```
   /TRACE CT,ON,COMP=ACTR
   /R 69,OPTIONS=(GROUPS((NETIF),(IUCV),(EZA),(CPT)),BUFFERS(256,64)),END
   ```

   You should see the following in the SYSLOG (Note the trace instance reported in the response. You will need it for subsequent commands):

   ```
   R 69,OPTIONS=(GROUPS((NETIF),(IUCV),(EZA),(CPT)),BUFFERS(256,64)),END
   IEE600I REPLY TO 069 IS;OPTIONS=(GROUPS((NETIF),(IUCV),(EZA),(CPT))
   T03TR909I Trace start successful Instance(01)
   ```

2. Enter the following command and response to update the ACTR trace with the limiting NETIF parameters for trace instance(1) on port 1350

   ```
   /TRACE CT,ON,COMP=ACTR
   /R 76,OPTIONS=(instance(1),GROUPS((NETIF,'PROTOCOL(TCP),PORT(1350),MDATA(16)'))),end
   ```

   You should see the following in the SYSLOG (Note the trace instance as you will need it for subsequent commands):

   ```
   R 76,OPTIONS=(INSTANCE(1),GROUPS((NETIF,'PROTOCOL(TCP),PORT(1350),
   MDATA(16)'))),END
   IEE600I REPLY TO 076 IS;OPTIONS=(INSTANCE(1),GROUPS((NETIF,'PROTOCO
   T03TR911I Trace modify successful Instance(01)
   ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND WERE
   SUCCESSFULLY EXECUTED.
   ```

3. Enter the following command and response to update the ACTR trace with the limiting IUCV parameters from job CICSPROD for trace instance(1)

   ```
   /TRACE CT,ON,COMP=ACTR
   /R 95,OPTIONS=(instance(1),GROUPS((IUCV,'USER(CICSPROD),MAXIUCVDATA(16)'))),end
   ```

   You should see the following in the SYSLOG (Note the trace instance as you will need it for subsequent commands):

   ```
   R 95,OPTIONS=(INSTANCE(1),GROUPS((IUCV,'USER(CICSPROD),MAXIUCVDATA(16)')
   )),END
   IEE600I REPLY TO 095 IS;OPTIONS=(INSTANCE(1),GROUPS((IUCV,'USER(
   T03TR911I Trace modify successful Instance(01)
   ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND WERE
   SUCCESSFULLY EXECUTED.
   ```

4. Enter the following command and response to update the ACTR trace with the limiting EZA parameters from job CICSPROD for trace instance(1)

   ```
   /TRACE CT,ON,COMP=ACTR
   /R 96,OPTIONS=(instance(1),GROUPS((EZA,'USER(CICSPROD),MAXEZADATA(16)'))),end
   ```

   You should see the following in the SYSLOG (Note the trace instance as you will need it for subsequent commands):

   ```
   R 96,OPTIONS=(INSTANCE(1),GROUPS((EZA,'USER(CICSPROD),MAXEZADATA(16)')))
   ,END
   IEE600I REPLY TO 096 IS;OPTIONS=(INSTANCE(1),GROUPS((EZA,'USER(
   T03TR911I Trace modify successful Instance(01)
   ```

5. Enter the following command and response to update the ACTR trace with the limiting CPT parameters from job CICSPROD for trace instance(1)

```
/TRACE CT,ON,COMP=ACTR
/R 112,OPTIONS=(instance(1),GROUPS((CPT,'USER(CICSPROD),TYPE(ACL,ACM,ADT,AFM,STR),
MAXCPTDATA(16)'))),end
```

   You should see the following in the SYSLOG (Note the trace instance as you will need it for subsequent commands):

```
R 112,OPTIONS=(INSTANCE(1),GROUPS((CPT,'TYPE(ACL,ACM,ADT,AFM,STR),MAXCPTDATA
(16)'))),END
IEE600I REPLY TO 112 IS;OPTIONS=(INSTANCE(1),GROUPS((CPT,'TYPE(ACL,
T03TR911I Trace modify successful Instance(01)
ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND WERE
SUCCESSFULLY EXECUTED.
```

Entering Trace Replies with a SYS1.PARMLIB Member

Here is a sample entry limiting tracing to port 1350 while performing NETIF, IUCV, EZA and CPT tracing. SYS1.PARMLIB(CTTCPWTR) member (CT trace parmlib members must start with the characters CT) trace containing the following entries:

```
TRACEOPTS
    ON
    OPTIONS(
    'GROUPS((NETIF,''PORT(1350),MDATA(16)''),
            (IUCV,''MIDATA(16)''),
            (EZA,''MEDATA(16)''),
            (CPT,''TYPE(ACL,ACM,ADT,AFM,STR),MCDATA(16)'')),
     BUFFERS(256,64),
     ')
```

The command to activate the PARMLIB member would be:

```
TRACE CT,ON,COMP=ACTR,PARM=CTTCPWTR
```

Please note that virtually any error you will encounter you will receive the following generic catch all error message:

```
T03TR903E Invalid OPTIONS specified
ITT004I START/STOP FAILED FOR TRACE COMP=ACTR 831
RETURN=00000004, REASON= 00000001.
ITT038I NONE OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND WERE
SUCCESSFULLY EXECUTED.
```

Among (but not all inclusive) the errors that you can receive the above message for include:

- Any syntax error:
- A misplaced () or comment
- A misspelled parameter
- An invalid parameter

- The OPTIONS field is too large (over 126 characters)
- If you experience this issue, you may have to consider placing the parameters into multiple parmlib members similar to the method described in section "Entering Long Trace Replies" earlier in this chapter.

So be aware to carefully review the entire command length since the combination of the parsing of the system console and the parsing of the actual trace command seems to not be at all tolerant of even the slightest simple syntax error and has a tendency to cover up many obvious errors with a generic error message reply.

Processing Trace Data

In order to process the data collected with the TRACE command, use the IPCS CTRACE command. It handles trace data that is in the MVS Component Trace Entry (CTE) format. By collecting trace data in this format, it is possible to use IPCS facilities to format the trace data.

Examine the SVCdump under IPCS (You may need to bring in the TCPaccess LINK library as a userlib library to allow IPCS to format the TCPEEP trace entries. Under IPCS the following command can be used to format the NETIF, IUCV, EZA and CPT trace records:

```
CTRACE SUMMARY COMP(ACTR) OPTIONS((FORMAT(TCP,DATA(16),IDATA(16),EZADATA(16),
CPTDATA(16)) GROUPS((NETIF,'PROTOCOL(TCP),MDATA(16)') (CPT) (EZA) (IUCV)))
```

Note: IPCS exit routines need to be written to locate trace records in dumps, filter trace records, and drive trace record formats.

Tracing TCP and Socket calls over IBM's TCPIP Stack

This section describes how to perform a CTRACE over the IBM TCPIP stack along with a TCPEEP to gather the following:

- CPT commands and responses   (From TCPEEP)
- EZASOKET calls and responses (From TCPEEP)
- EZASMI calls and responses (From Component Trace)
- TCP packets and data (From Component Trace)

To gather the EZASMI calls, TCP packets, and data over IBM's TCPIP stack requires that a site runs the Component trace utility. For more information, see IBMs APAR II12014 and *z/OS Communications Server IP Diagnosis Guide Version 1 Release 5 (GC31-8782-04 or f1a1c530.boo).*

How to set up a Component Trace File

At our site we have set up a procedure in SYS1.PROCLIB called NMDWTR to write to a file called SYSPROG.CA11.NMD.CTRACE. One must have a component trace file available with an adequate space allocation. No secondary quantity should be defined to insure the trace will wrap successfully. Such an allocation might be:

```
SYSPROG.CA11.NMD.CTRACE -------------
Organization  . . . : PS
Record format . . . : VB
Record length . . . : 32756
Block size  . . . . : 32760
1st extent cylinders: 300
Secondary cylinders : 0
```

Although many sites use files where the record length is 27994 and block size of 27998.

Here is an example writer proc NMDWTR in dataset SYS1.PROCLIB:

```
//NMDWTR  PROC
//NMDWTR  EXEC PGM=ITTTRCWR,REGION=32M
//TRCOUT01  DD DISP=SHR,DSN=SYSPROG.&SYSNAME..NMD.CTRACE
```

Activating the Component Trace for the IBM TCPIP Stack

In this example, we are going to trace activity associated with a CICS job called CICSPROD over an IBM TCPIP stack called TCPIP11. We will be collecting both the SOCKAPI calls and TCP level information for sessions running over port 1350.

1. Start the External Writer procedure in wrap mode

   /TRACE CT,WTRSTART=NMDWTR,WRAP

2. Start the SYSTCPIP event trace and make the following responses

   /TRACE CT,ON,COMP=SYSTCPIP,SUB=(TCPIP11)

   /R xx,JOBNAME=(CICSPROD),OPTIONS=(sockapi),CONT

   /R xx,WTR=NMDWTR,END

3. Start the SYSTCPDA packet trace and make the following responses

   /TRACE CT,ON,COMP=SYSTCPDA,SUB=(TCPIP11)

   /R xx,WTR=NMDWTR,END

4. Turn on the packet trace with a vary command

   /V TCPIP,TCPIP11,PKT,ON

   /V TCPIP,TCPIP11,PKT,DEST=1350

   /V TCPIP,TCPIP11,PKT,SRCP=1350

5. Turn on the data tracing

   /V TCPIP,TCPIP11,DAT

Turn on TCPEEP for CPT Tracing

The following TCPEEP job to capture CPT TCPEEP data where the trace subsystem name is ACTR:

```
//TCPEEP   JOB ...
//*
//TCPEEP   EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=4M,PARM=
//STEPLIB  DD DISP=SHR,DSN=NTWRKIT.V600QA.LINK
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*,LRECL=132,BLKSIZE=132,RECFM=FB
//SYSIN    DD DUMMY
//*
```

```
//SYSTSIN  DD *
 TCPEEP TRCSSID(ACTR)                                                  -
 FORMAT(TCP,DATA(9999),IDATA(9999),EZADATA(9999),CPTDATA(9999))        -
 GROUPS((NETIF,'PROTOCOL(TCP),MDATA(9999)'),                           -
        (CPT,'TYPE(ACL,ACM,ADT,AFM,AFT,AXL,STR,QUE),MCDATA(9999)'),    -
        (EZA,'MEDATA(9999)'),                                          -
        (IUCV,'MIDATA(9999)'))
/*
```

Note: The data parameters in the above JCL can be lowered from 9999 to 64 in cases where the issue does not involve a data integrity issue.

Recreate the Problem Scenario

At this point all the tracing has been enabled. A site should recreate the problem scenario.

After the problem scenario has been recreated one should turn off all the traces so they don't wrap. Do the following:

1. Turn off the packet trace

   /V TCPIP,TCPIP11,PKT,OFF

2. Turn off the data trace

   /V TCPIP,TCPIP11,DAT,OFF

3. Turn off the SYSTCPDA packet trace

   /TRACE CT,OFF,COMP=SYSTCPDA,SUB=(TCPIP11)

4. Turn off the SYSTCPIP event trace

   /TRACE CT,OFF,COMP=SYSTCPIP,SUB=(TCPIP11)

5. Stop the writer

   /TRACE CT,WTRSTOP=NMDWTR,FLUSH

Send the TRS Form of the CTRACE file and the TCPEEP File to CA

We need both the TCPEEP job output and the TRS form of the CTRACE file. Tell us the DCB attributes used for the original CTRACE file and TCPEEP file.

One must use the TRSMAIN utility with the PACK option to create a file that can be transferred on and off of MVS in binary while retaining the file's data integrity. The correct sequence of events for transferring a CTRACE file from one MVS site is to:

1. Create a CTRACE file.

2. Use TRSMAIN with the PACK parm option to create a TRS file. Create a file with DCB attributes PS, RECFM=FB, LRECL=1024, BLKSIZE=6144.

3. FTP in binary the TRS file to a non MVS site. There may be a number of binary transfers at this point.

4. FTP in binary the TRS file back to MVS. Create a file with DCB attributes PS, RECFM=FB, LRECL=1024, BLKSIZE=6144.

5. Use TRSMAIN with the UNPACK parm option to create a CTRACE file. Make sure you use the same dcb attributes as the original DCB attributes.

Here is sample TRSMAIN JCL to create a TRS file (HUSJO02.CA11.NMD.CTRACE.TRS) from a CTRACE file (HUSJO02.CA11.NMD.CTRACE):

```
//STEP1    EXEC PGM=TRSMAIN,PARM=PACK
//STEPLIB  DD DISP=SHR,DSN=SYS2.LINKLIB
//SYSPRINT DD SYSOUT=*
//*
//INFILE   DD DISP=SHR,DSN=HUSJO02.CA11.NMD.CTRACE
//*
//OUTFILE  DD DSN=HUSJO02.CA11.NMD.CTRACE.TRS,
//*        DISP=(,CATLG),
//         STORCLAS=NMDPOOL,
//         SPACE=(CYL,(10,100),RLSE),
//         DCB=(RECFM=FB,LRECL=1024,BLKSIZE=6144)
```

Here is some sample TRSMAIN JCL to unpack a TRS file (HUSJO02.CA11.NMD.CTRACE.TRSBACK) into a CTRACE file (HUSJO02.CA11.NMD.CTRACE.UNPACK):

```
//STEP1    EXEC PGM=TRSMAIN,PARM=UNPACK
//STEPLIB  DD DISP=SHR,DSN=SYS2.LINKLIB
//SYSPRINT DD SYSOUT=*
//*
//INFILE   DD DISP=OLD,DSN=HUSJO02.CA11.NMD.CTRACE.TRSBACK
//*
//OUTFILE  DD DSN=HUSJO02.CA11.NMD.CTRACE.UNPACK,
//         DISP=(,CATLG),
//         STORCLAS=NMDPOOL,
//         SPACE=(CYL,(20,),RLSE),
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
```

Useful IPCS CTRACE Commands to Format the Data

In this section we have listed a number of useful CTRACE commands for formatting the CTRACE data from the CICS region called CICSPROD and TCPIP job called TCPIP11:

Follow the API calls:

```
CTRACE COMP(SYSTCPIP) SUB(TCPIP11)) FULL JOBNAME(CICSPROD)
```

For TCP packets in time:

```
CTRACE COMP(SYSTCPDA) SUB(TCPIP11)) LOCAL FULL  SUM
```

Very useful for non local host sessions:

```
CTRACE COMP(SYSTCPDA) SUB(TCPIP11)) LOCAL FULL
OPTIONS((SESSION TCP))
```

Break apart the TCP packet:

```
CTRACE COMP(SYSTCPDA) SUB(TCPIP11)) FULL LOCAL
OPTIONS((FORMAT))
```

See the Data in the TCP packets:

```
CTRACE COMP(SYSTCPDA) SUB(TCPIP11)) FULL LOCAL
OPTIONS((DUMP))
```

See a report for when a session starts and ends:

```
CTRACE COMP(SYSTCPDA) SUB(TCPIP11)) FULL LOCAL
OPTIONS((REASSEMBLY(DETAIL) STAT))
```

MVS DUMP and SLIP SET Commands

When the IFS SVCD command is used with Unicenter TCPaccess, IFSPDUMP internally builds the command to dump the data space containing our trace records. However, data spaces are not dumped by default.

When using the SLIP SET or DUMP commands, you must explicitly dump the data space by including the DSPNAME parameter in the SLIP SET or DUMP command string. The DSPNAME parameter is either *asid.name* or '*jobname*'.*name*. For example, if the TRACE job name is SNSTRACE, you would code DSPNAME=('SNSTRACE'.INLKTRCE).

Note: The single quotes (' ') bracketing the job name are required. INLKTRCE is the name used internally to identify the data space.

Example: MVS DUMP Command

Here is an example of using the MVS DUMP command and responses to dump CICS, Unicenter TCPaccess, and TRACE address spaces:

```
/dump title=('dump qats22r9')
*048 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND

/48jobname=(qats22r9,dcltcp60,dcltrace),dspname=('dcltrace'.inlktrce),cont
*049 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND

/49sdata=(CSA,LPA,LSQA,PSA,RGN,SQA,SUM,TRT),end
```

Example: MVS SLIP SET Command

Here is an example of the SLIP SET syntax needed to do the same thing in the event of an abend:

```
/slip set,id=myid,comp=0cx,j=qats22r9,action=svcd
,jl=(qats22r9,dcltcp60,dcltrace)
,dspname=('dcltrace'.inlktrce)
,matchlim=5,SD=(CSA,LPA,LSQA,PSA,RGN,SQA,SUM,TRT),END
```

Entering these commands at the console would look like this:

```
/slip set,id=myid,comp=0c1,j=qats22r9,action=svcd
*119 IEE726D ENTER ADDITIONAL SLIP PARAMETERS, 'END', OR 'CANCEL'

/119,jl=(qats22r9,dcltcp60,dcltrace)
*120 IEE726D ENTER ADDITIONAL SLIP PARAMETERS, 'END', OR 'CANCEL'

/120,dspname=('dcltrace'.inlktrce)
*121 IEE726D ENTER ADDITIONAL SLIP PARAMETERS, 'END', OR 'CANCEL'

/121,matchlim=5,SD=(CSA,LPA,LSQA,PSA,RGN,SQA,SUM,TRT),END
IEE600I REPLY TO 121 IS;MATCHLIM=5,SD=(CSA,LPA, LSQA,PSA,RGN,SQA,SUM
IEE727I SLIP TRAP ID=MYID SET
```

Chapter

8 Troubleshooting

This chapter contains information about identifying and resolving problems.

The following topics are discussed:

- Diagnostic Procedures
- Accessing the Online Client Support System
- CA-TLC: Total License Care
- Contacting Technical Support
- Generating a Problem Report
- Product Versions and Maintenance
- Requesting Enhancements

Diagnostic Procedures

The following sections detail the procedures you should follow if you have a problem with a Computer Associates software product.

Collecting Diagnostic Data

In the table below, use the left column to categorize the problem your site encountered. Then, follow the instructions in the corresponding right column to generate useful diagnostic data.

Type of Problem	Procedure
What to Do if the NetMaster Command Processor Abnormally Terminates	Under normal circumstances the NetMaster Command Processor abnormal termination should not occur. In the unlikely situation that it does, problem determination and diagnostic information must be retained to enable Computer Associates engineers to troubleshoot the problem. To ensure the problem is resolved as quickly as possible, the following diagnostic materials (if available) must be forwarded to Computer Associates technical support: ■ The T09TCMDS dump (written to a CICS dump data set) ■ System and CICS/TS message logs ■ An SVC dump of the CICS region **Note:** If the problem can be reproduced, supply a TCPEEP trace of TCP traffic over the NetMaster Command Processor port leading up to the problem.
EZASOKET or EZACICAL call failing with bad Errno field	Refer to IBM's *IP CICS Socket's Guide* and check the listing for all messages. Save all system output. If the errno message itself does not suggest what diagnostic approach to take please follow the procedure Debugging Guidelines for a Unicenter SOLVE:CPT Application.
Installation	Refer to the install steps in *Getting Started*. Save all output.
Unicenter SOLVE:CPT Error Message	Review message in the *Message Guide*. If the message itself does not suggest which diagnostic approach to take, follow the procedure provided in Debugging Guidelines for a Unicenter SOLVE:CPT Application.

Debugging Guidelines for a Unicenter SOLVE:CPT Application

Throughout this section, the term *EZASOKET* can represent either an EZASOKET or EZACICAL application.

Note: Customer's with CPT, EZASOKET, or EZACICAL API failures can debug their application by gathering the following information **all at one time**. Please do not send CA support pieces run from separate sessions as the time stamps will not match up between components, making a valid analysis impossible.

Follow the steps below to provide CA customer support with sufficient information to diagnose your issue. References to other sections are interspersed when needed for further detail.

Preparation Steps

1. Refer to Step 1: Debugging Guidelines Required Maintenance PTFs for possible maintenance requirements needed to perform the described diagnostics.
2. **Unicenter TCPaccess customers only:** Follow instructions in Step 2: Raise the Size of Unicenter TCPaccess' Internal IFS (Below the Line) Trace Table.
3. Contact the vendor of the failing CPT and/or EZASOKET API application for specific debugging procedures available to their product.
4. Bring up component trace address space first and then CICS/TS. The member T09TRACE in the T09SAMP data set is an example set of JCL for the Component trace subsystem address space. See the Trace section in the "Diagnostic Commands" chapter for detailed installation and startup information.
5. Unicenter TCPaccess customers only: Issue MVS operator command to the TCPaccess address space to list PC and CSA module addresses in the JES logs (if the Unicenter TCPaccess jobname is TCPIP):

   ```
   F TCPIP,MODULE ALL
   ```
6. Start TCPEEP trace display formatting by following instructions in Step 6 Part I: Required TCPEEP Command Examples.
7. Start Unicenter SOLVE:CPT.
8. Start the application and wait for the hang or failure to occur.

When the Problem (abend or hang) occurs:

9. **Unicenter TCPaccess customers only:** Issue MVS operator command to the TCPaccess address space to dump and format API control blocks. If the TCPaccess job name is TCPIP:

   ```
   F TCPIP,TCP SNAP ALL
   ```
10. Follow instructions in Step 10 Part I: Take SVC Dumps of ALL the Related API Application Address Spaces.

11. Send CA support **everything**:
    - All of the TCP/IP job output including JES logs
    - All of the API application job output including JES logs
    - SVC dumps of the TCP/IP, CICS/TS, and optionally T09TRACE

      T09TRACE is required if your application falls in the description in Step 6 Part II: Hard to Recreate (Long Running) Traces
    - Make sure you include the DCB attributes of the original SVC dump.
    - TCPEEP output
    - **Unicenter TCPaccess customers only:** TCP SNAP ALL output from the TCPaccess address space

Important! *Do not send support pieces run from separate sessions as the time stamps do not match up between components and a valid analysis is simply impossible.*

Step 1: Debugging Guidelines Required to Maintain PTFs

Unicenter TCPaccess required PTFs:

- The exact same procedures are used to debug IUCV or HPNS applications at the 5.2 level or above.
- EZA trace support was introduced at the 5.2 level of Unicenter TCPaccess with PTF TP08600. EZA trace support was introduced at the 5.3 level of Unicenter TCPaccess with PTF TP08599.

Step 2: Raise the Size of Unicenter TCPaccess Internal IFS (Below the Line) Trace Table

This keeps the IFS internal trace table from wrapping too quickly. One can change the startup parameters or dynamically reallocate Unicenter TCPaccess address space internal trace table using the TRACE operator command.

In the Unicenter TCPaccess startup JCL, there is a CMND parameter representing a member name in the SYSPROC DD library. You can set the IFS trace table to one MB by placing the following command in the CMND PDS member:

```
MODIFY TRACE ON SIZE(256)
```

To use one-half megabyte of above the line storage for the IFS trace table, set SIZE(128) on the MODIFY command.

You can dynamically reallocate Unicenter TCPaccess address space internal trace table using the TRACE operator command. Refer to the TRACE section under IJT commands in the *Unicenter® TCPaccess™ Communications Server System Management Guide,* which documents the TRACE operator command.

A sample command to reallocate a large IFS trace table:

```
MODIFY TRACE ON SIZE(256)
```

Step 6 Part I: Required TCPEEP Command Examples

There are issues where it may take a long time to re-create a problem. Under these circumstances, long running TCPEEP traces can consume extremely large amounts of the JES SPOOL. The Unicenter SOLVE:CPT trace address space can buffer large amounts of data inside the trace address space when using the MVS console operator command: TRACE CT.

If this condition applies to your situation, proceed to Step 6 Part II: Hard to Recreate (Long Running) Traces otherwise use the following instructions.

You can use TCPEEP to trace data, commands, and responses through CPT, EZASOKET and out onto the network.

Start TCPEEP with the following for a TCP problem (limiting data to 16 bytes for issues where we are unconcerned with the data):

```
TCPEEP TRCSSID(ACTR)                                          +
GROUPS((NETIF),(IUCV,'MAXIUCVDATA(16)'),                      +
       (EZA,’MAXEZADATA(16)'),                                +
       (CPT,'TYPE(ACL,ACM,ADT,AFM,STR),MAXCPTDATA(16)'))      +
FORMAT(TCP,DATA(16),IUCVDATA(16),EZADATA(16),CPTDATA(16))
```

If running an UDP application, use the UDP invocation option.

```
TCPEEP TRCSSID(ACTR)                                          +
GROUPS((NETIF),(IUCV,'MAXIUCVDATA(16)') ,                     +
       (EZA,'MAXEZADATA(16)'),                                +
       (CPT,'TYPE(ACL,ACM,ADT,AFM,STR),MAXCPTDATA(16)'))      +
FORMAT(UDP,DATA(16),IUCVDATA(16),EZADATA(16),CPTDATA(16))
```

You can limit TCP or UDP trace output when you invoke TCPEEP with the PORT(), HOST(), and PROTOCOL() sub parameters on the GROUPS NETIF filter parameter. Refer to the "Diagnostic Commands" chapter in the *Unicenter TCPaccess Communications Server System Management Guide* for full information on limiting parameters.

Most problem issues can be associated with a server port and an application address space. Data integrity is not a key issue most problem issues, so one can greatly limit TCPEEP job output by limiting the amount of data printed and collected on the associated TCPEEP data parameters

In the following example, TCPEEP traces TCP data for host IP address 10.12.14.32 and for batch job CICSPROD and the first 32 bytes of EZASOKET, HPNS and TCP data for an issue without data integrity issues:

```
TCPEEP TRCSSID(ACTR) +
       FORMAT(TCP,DATA(32),IUCVDATA(32), EZADATA(32),CPTDATA(32))     +
       GROUPS((NETIF,'PROTOCOL(TCP),HOST(10.12.14.32),MAXDATA(32)'),  +
              (IUCV,'USER(CICSPROD),MAXIUCVDATA(32)'),                +
              (EZA,'MAXEZADATA(32)'),                                 +
              (CPT,'TYPE(ACL,ACM,ADT,AFM,STR),MAXCPTDATA(32)' ))
```

In an atypical problem where a user truly needs to sort out a problem with data integrity, a server using port 2151 in address space named CICSPROD can use these TCPEEP parameters:

```
TCPEEP TRCSSID(ACTR) +
       FORMAT(TCP,DATA(32000),IUCVDATA(32000), EZADATA(32000))       +
       GROUPS((NETIF,'PROTOCOL(TCP), PORT(2151), MAXDATA(32000)'),   +
              (IUCV,'USER(CICSPROD),MAXIUCVDATA(32000)'),            +
              (EZA,'MAXEZADATA(32000)' ),                            +
              (CPT,'TYPE(ACL,ACM,ADT,AFM,STR),MAXCPTDATA(32000)' ))
```

Important! *With the data amount set very high such as 32000 above, a very large trace is generated in a short period. This high a data value should be avoided unless you are absolutely convinced that there is a data integrity problem. For the vast majority of problems a glance value of 16 is sufficient. A value of 16 causes one line of data to display.*

Step 6 Part II: Hard to Recreate (Long Running) Traces

There are issues where it may take a long time to recreate a problem. Under these circumstances, long running TCPEEP traces can consume extremely large amounts of the JES SPOOL. The Unicenter SOLVE:CPT trace address space can buffer large amounts of data inside the trace address space when utilizing the MVS console operator command: TRACE CT.

Note: It is very important to stop the trace address space from wrapping by dumping the trace address space as close to the point of failure as possible. A few minutes of time on a busy system could easily overlay the needed trace information by wrapping.

Issue the following MVS console operator command to turn on system CT tracing where the subsystem of the trace address space is ACTR:

```
/TRACE CT,ON,COMP=ACTR
```

After the previous command is entered, the operator is prompted by Unicenter SOLVE:CPT trace to enter the appropriate tracing related variables. Enter the GROUPS and BUFFERS related parameters in the OPTIONS field.

The GROUPS parameters are the exact same format as the GROUPS parameters in a TCPEEP command.

The BUFFERS (SIZE,NUMBER) variable in the trace address space allocates storage for the tracing. Setting BUFFERS to a large SIZE and NUMBER will reserve more storage for tracing.

- SIZE is in kilobytes and ranges from 64 to 1024
- NUMBER ranges from 2 to 128

In the following example, the trace address space gathers TCP data for port 3666; HPNS, CPT and EZA sockets API tracing for user ID CICSPROD; and the first 16 bytes of HPNS and EZA sockets API data:

```
R 82,OPTIONS=(GROUPS((NETIF,'PROTOCOL(TCP),PORT(3666),
MDATA(16)'),(IUCV,'USER(CICSPROD),MAXIUCVDATA(16)')),(
EZA,'USER(CICSPROD),MAXEZADATA(16)’),(CPT,'USER(CICSPR
OD),TYPE(ACL,ACM,ADT,AFM,STR),MAXCPTDATA,(16)')),BUFFE
RS(256,64)),END
```

At the point, when the failure occurs you must take an SVCDUMP of the trace address space to gather the trace data ***before*** the trace internally wraps. Refer to the SVCDUMP command discussion in Step 10 Part I: Take SVC Dumps of ALL the Related API Application Address Spaces to learn how to take an SVCDUMP of more than one address space.

The following command under IPCS for the SVC dump output can format the data using the GROUP and FORMAT parameters (the same exact parameter formats as those used with TCPEEP) are placed in the OPTIONS field where the subsystem of the trace address space is ACTR:

```
CTRACE SUMMARY COMP(ACTR) OPTIONS((FORMAT(TCP,DATA(16),IUCVDATA(16),EZADATA(16),CP
TDATA(16)),GROUPS((NETIF,'PROTOCOL(TCP),MAXDATA(16)'),(IUCV,'MAXIUCVDATA(16)'),(EZ
A,’MAXEZADATA(16)’),(CPT,’TYPE(ACL,ACM,ADT,AFM,STR),MAXCPTDATA(16)’)))
```

The IPCS CTRACE TALLY command gives you totals for each tracing record type when the tracing subsystemid is ACTR:

```
CTRACE TALLY  COMP(ACTR)
```

Entering Long Trace Replies

There are cases in which you want to create a high level of filtering in order to limit the size of the trace data being collected. It is an MVS restriction to limit the reply to the trace prompt to 126 characters in total length. The best way around this restriction is to enter and modify the original trace instance. The following set of operator commands is equivalent of entering the following trace options reply all at once if you were not limited to entering 126 characters on a command line:

```
/R 82,OPTIONS=(GROUPS((NETIF,'PROTOCOL(TCP),PORT(1350),MDATA(16)'),
(CPT,'USER(CICSPROD),TYPE(ACL,ACM,ADT,AFM,STR),MCPTDATA(16)’),(EZA,'USER(CICSPROD)
,MEZADATA(16)'),(IUCV,'USER(CICSPROD),MIDATA(16)')),BUFFERS(256,64)),END
```

In the example the following is assumed:

- The trace subsystem is ACTR.
- The TCPaccess job is called RUNTCP.
- The Trace address space job is called RUNTRACE
- The CICS job is call CICSPROD.
- The only port we are interested in tracing is port 1350.
- The trace instance is 1 after the first trace operator command.
- The CPT trace entries we are interested in are for CLOSE (ACL), LISTEN (ACM), SEND and RECEIVE (ADT), GIVE and TAKE events (AFM) and started task events (STR).

6. Enter the following command and response to get the CT trace going for tracing subsytem ACTR for NETIF, IUCV, EZA and CPT tracing with a large buffer pool for tracing:

   ```
   /TRACE CT,ON,COMP=ACTR
   /R 69,OPTIONS=(GROUPS((NETIF),(IUCV),(EZA),(CPT)),BUFFERS(256,64)),END
   ```

   You should see the following in the SYSLOG (Note the trace instance reported in the response. You will need it for subsequent commands):

   ```
   R 69,OPTIONS=(GROUPS((NETIF),(IUCV),(EZA),(CPT)),BUFFERS(256,64)),END
   IEE600I REPLY TO 069 IS;OPTIONS=(GROUPS((NETIF),(IUCV),(EZA),(CPT))
   T03TR909I Trace start successful Instance(01)
   ```

7. Enter the following command and response to update the ACTR trace with the limiting NETIF parameters for trace instance(1) on port 1350:

```
/TRACE CT,ON,COMP=ACTR
/R 76,OPTIONS=(instance(1),GROUPS((NETIF,'PROTOCOL(TCP),PORT(1350),MDATA(16)'))),end
```

   You should see the following in the SYSLOG (Note the trace instance as you will need it for subsequent commands):

```
R 76,OPTIONS=(INSTANCE(1),GROUPS((NETIF,'PROTOCOL(TCP),PORT(1350),
MDATA(16)'))),END
IEE600I REPLY TO 076 IS;OPTIONS=(INSTANCE(1),GROUPS((NETIF,'PROTOCO
T03TR911I Trace modify successful Instance(01)
ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND WERE
SUCCESSFULLY EXECUTED.
```

8. Enter the following command and response to update the ACTR trace with the limiting IUCV parameters from job CICSPROD for trace instance(1):

```
/TRACE CT,ON,COMP=ACTR
/R 95,OPTIONS=(instance(1),GROUPS((IUCV,'USER(CICSPROD),MAXIUCVDATA(16)'))),end
```

   You should see the following in the SYSLOG (Note the trace instance, as you will need it for subsequent commands):

```
R 95,OPTIONS=(INSTANCE(1),GROUPS((IUCV,'USER(CICSPROD),MAXIUCVDATA(16)')
)),END
IEE600I REPLY TO 095 IS;OPTIONS=(INSTANCE(1),GROUPS((IUCV,'USER(
T03TR911I Trace modify successful Instance(01)
ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND WERE
SUCCESSFULLY EXECUTED.
```

9. Enter the following command and response to update the ACTR trace with the limiting EZA parameters from job CICSPROD for trace instance(1):

```
/TRACE CT,ON,COMP=ACTR
/R 96,OPTIONS=(instance(1),GROUPS((EZA,'USER(CICSPROD),MAXEZADATA(16)'))),end
```

   You should see the following in the SYSLOG (Note the trace instance, as you will need it for subsequent commands):

```
R 96,OPTIONS=(INSTANCE(1),GROUPS((EZA,'USER(CICSPROD),MAXEZADATA(16)')))
,END
IEE600I REPLY TO 096 IS;OPTIONS=(INSTANCE(1),GROUPS((EZA,'USER(
T03TR911I Trace modify successful Instance(01)
```

10. Enter the following command and response to update the ACTR trace with the limiting CPT parameters from job CICSPROD for trace instance(1):

```
/TRACE CT,ON,COMP=ACTR
/R 112,OPTIONS=(instance(1),GROUPS((CPT,'USER(CICSPROD),TYPE(ACL,ACM,ADT,AFM,STR),
MAXCPTDATA(16)'))),end
```

You should see the following in the SYSLOG (Note the trace instance, as you will need it for subsequent commands):

```
R 112,OPTIONS=(INSTANCE(1),GROUPS((CPT,'TYPE(ACL,ACM,ADT,AFM,STR),MAXCPTDATA
(16)'))),END
IEE600I REPLY TO 112 IS;OPTIONS=(INSTANCE(1),GROUPS((CPT,'TYPE(ACL,
T03TR911I Trace modify successful Instance(01)
ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND WERE
SUCCESSFULLY EXECUTED.
```

Step 10 Part I: Take SVC Dumps of *ALL* the Related API Application Address Spaces

CA support needs SVC dumps of **all** the related API application address spaces including the CICS/TS region; the TCP/IP address space; and optionally the trace component address space.

Note: If you meet the criteria in Step 6 Part II: Hard to Recreate (Long Running) Traces, then it is required that you also SVC dump the trace component address space as described below.

Refer to IBM manual *MVS System Commands* as the definitive source for both the SLIP and DUMP MVS operator commands.

In an **abend situation,** use the instructions in Step 10 Part II: SLIP Command for an Abend Situation

In a **hung application situation**, use the instructions in one of the following sections depending on the stack you are using (CA or IBM TCP/IP):

- Step 10 Part III: (Unicenter TCPaccess Only) SVCDUMP Command for a Hung Application Situation command for a hung application situation
- Step 10 Part IV: (IBM TCP/IP Only) SVCDUMP Command for a Hung Application Situation for a hung application situation

Step 10 Part II: SLIP Command for an Abend Situation

The JOBLIST or ASIDLST parameters on the SLIP MVS operator command enables you to take a SVCDUMP of multiple address spaces when a single address space is going to abend. In our example below, we use the JOBLIST (or JL) parameter to dump all the API related address spaces:

- CICSPROD
- Trace address space (jobname T09TRACE)
- The TCP/IP stack (jobname of TCPIP)
- Each abend situation is different, set your slip trap for the abend address space at your site

When using the MVS SLIP SET command, you must explicitly dump the T09TRACE data space by including the DSPNAME parameter in the SLIP SET command string. The DSPNAME parameter is either *asid.name* or '*jobname*'.*name*. For example, if the TRACE job name is T09TRACE, you would code DSPNAME=('T09TRACE'.INLKTRCE).

Note: The single quotes (' ') bracketing the jobname are required. INLKTRCE is the name used internally to identify the data space.

In our example, address space CICSPROD abends with a S0C1 so the JOBNAME (abbreviated J) parameter must be set to CICSPROD.

Make sure we get CSA, all the private region, system trace, registers, RTM2WA, SQA, and LPA. Do this by setting SDATA parameters to:

```
CSA LPA LSQA PSA RGN SQA SUM TRT
```

You can use the following SLIP command to SVC dump the CICSPROD, trace, data space, and TCP/IP address spaces when CICSPROD takes a S0C1 abend:

```
/SLIP SET,ID=MYID,COMP=0C1,J=CICSPROD,ACTION=SVCD
,JL=(CICSPROD,TCPIP,T09TRACE)
,DSPNAME=('T09TRACE'.INLKTRCE)
,MATCHLIM=1,SD=(CSA,LPA,LSQA,PSA,RGN,SQA,SUM,TRT),END
```

Entering this command and responses at the console would look like this:

```
/SLIP SET,ID=MYID,COMP=0C1,J=CICSPROD,ACTION=SVCD *119 IEE726D ENTER ADDITIONAL
SLIP PARAMETERS, 'END', OR 'CANCEL'

/119,JL=(CICSPROD,TCPIP,T09TRACE)
*120 IEE726D ENTER ADDITIONAL SLIP PARAMETERS, 'END', OR 'CANCEL'

/120, ,DSPNAME=('T09TRACE'.INLKTRCE)
*121 IEE726D ENTER ADDITIONAL SLIP PARAMETERS, 'END', OR 'CANCEL'

/121, MATCHLIM=5,SD=(CSA,LPA,LSQA,PSA,RGN,SQA,SUM,TRT),END
IEE600I REPLY TO 121 IS;MATCHLIM=5,SD=(CSA,LPA, LSQA,PSA,RGN,SQA,SUM
IEE727I SLIP TRAP ID=MYID SET
```

Important! *Make sure that you can tell support the LRECL, RECFM, and BLKSIZE DCB attributes of the original SVC dump.*

Step 10 Part III: (Unicenter TCPaccess Only) SVCDUMP Command for a Hung Application Situation

Unicenter TCPaccess SVCDUMP operator command enables a user to dump multiple address spaces when a hang occurs.

You can dump multiple address spaces by specifying either:

- JOBNAME(<*jobname1*>,<*jobname2*>,...)
- ASID(<asid1>,<asid2>,...) parameters.

If you needed to dump three address spaces:

- TCPaccess (job name TCPIP)
- Trace address space (job name T09TRACE)
- Job name CICSPROD

Issue the following operator command:

```
F TCPIP,SVCDUMP JOBNAME(CICSPROD,T09TRACE)
```

Important! *Make sure that you can tell support the LRECL, RECFM, and BLKSIZE DCB attributes of the original SVC dump.*

Step 10 Part IV: (IBM TCP/IP Only) SVCDUMP Command for a Hung Application Situation

If you are using standard MVS commands, follow the sequence below.

When using the MVS DUMP command, you must explicitly dump the T09TRACE data space by including the DSPNAME parameter in the SLIP SET command string. The DSPNAME parameter is either *asid.name* or '*jobname*'.*name*. For example, if the TRACE job name is T09TRACE, you would code DSPNAME=('T09TRACE'.INLKTRCE).

Note: The single quotes (' ') bracketing the jobname are required. INLKTRCE is the name used internally to identify the data space.

Important! *Note that matching the SDATA below exactly is critical to Computer Associates Support having adequate information to help you with your issue.*

If you need to dump three address spaces:

- TCP/IP (job name TCPIP)
- Trace address space (job name T09TRACE)
- Job name CICSPROD

Use the following sequence of operator commands:

Note: *Itialized data* is site-dependent.

1. Enter: DUMP COMM=(my socket app name hang in CICS)
2. System response:
   ```
   *98 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
   ```
3. Enter:
   ```
   R 98,JOBNAME=(CICSPROD,TCPIP,T09TRACE),DSPNAME=('T09TRACE'.INLKTRC),CONT
   ```
4. System response:
   ```
   IEE600I REPLY TO 98
   IS;JOBNAME=(CICSPROD,TCPIP,T09TRACE),DSPNAME=('T09TRACE'.INLKTRC),CONT

   *99 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
   ```
5. Enter: R *99*,SDATA=(CSA,LPA,LSQA,PSA,RGN,SQA,SUM,TRT),END
6. System response:
   ```
   IEE600I REPLY TO 99
   IS;SDATA=(CSA,LPA,LSQA,PSA,RGN,SQA,SUM,TRT),END
   IEA794I SVC DUMP HAS CAPTURED: 237
   DUMPID=126 REQUESTED BY JOB (*MASTER*)
   DUMP TITLE=my socket app name hang in CICS
   ```

Important! *It is very important that you tell us the LRECL, RECFM, and BLKSIZE DCB attributes of the original SVC dump.*

Interpreting Diagnostic Data

Once you have collected the specified diagnostic data, write down your answers to the following questions:

1. What was the sequence of events before the error condition?
2. What circumstances existed when the problem occurred and what action did you take?
3. Has this situation occurred before? What was different then?
4. Did the problem occur after a particular PTF was applied or after a new release of the software was installed?
5. Have you recently installed a new release of the operating system?
6. Has the hardware configuration (tape drives, disk drives, and so forth) changed?

From your response to these questions and the diagnostic data, try to identify the cause and resolve the problem.

Accessing the Online Client Support System

Computer Associates is making extensive use of the Internet for your benefit. CA encourages you to *surf the net* to the CA home page at *ca.com* and the support site at http://supportconnect.ca.com. The CA Internet site provides a great variety of information about CA products and services, including:

- Service and support
- Product information and sales
- CA-World conference information
- Press releases
- CA user groups

StarTCC, the web-based portion of CA-TCC (CA-Total Client Care) gives you real time, interactive access to CA product support information through the Internet.

Using StarTCC, you can:

- Open new issues
- Browse or update your existing issues and enhancement requests
- Perform keyword searches
- Download solutions, PTFs, and important notices regarding CA products, maintenance, and documentation

Requirements for Using StarTCC

The following are the requirements for using StarTCC:

- You must be a CA client with a current maintenance agreement.
- You must register through the CA Internet site.
- You must access the Internet with a browser that supports the HTML specification 2.0 or higher, such as Netscape Navigator 2.0 or higher or Microsoft Internet Explorer 3.0 or higher.
- Browsers that meet the HTML requirement support the following functions, which are required for StarTCC:
  - Secure sockets layer (SSL) to encrypt your transaction traffic
  - Encrypted data records (known as COOKIES)
  - HTML tables

StarTCC Security

StarTCC runs as a secured server (SSL). You may need to configure your browser to enable SSL. Guidelines for doing this are provided on the CA Technical Support page.

Accessing StarTCC

To access StarTCC, go to http://supportconnect.ca.com. The StarTCC options are:

- StarTCC Information
- StarTCC Registration
- Access StarTCC

These options are described below.

StarTCC Information: Select the information option to view background information for StarTCC, details about the prerequisites, and instructions for configuring your browser. Be sure to review this section for updates or information not included here.

StarTCC Registration: Select the registration option to identify yourself to StarTCC. You must register before you can access StarTCC online. There are prompts for all required information, including your name, site ID, CA-StarTrak PIN, company name, E-Mail address, postal address, and desired password for accessing StarTCC.

Note: If you do not have a CA-StarTrak PIN, StarTCC provides one for you when you register.

Access StarTCC: Select the access option to begin using StarTCC. When prompted, enter your user ID and password. Once your sign-on is validated, you can perform the following:

- Open a new issue
- Open an issue for, or request an enhancement to, one of your CA products.
- Browse your issues and enhancement requests
- Display all issues for your site. The issues are grouped into three categories:

  Open, Closed, and Enhancement Requests (DARs).

  Browse and/or download solutions
- Specify criteria for selecting solutions, which you can then view or download.
- Search the CA knowledge base
- Specify criteria for searching the CA database for solutions, problems, and keywords that can provide you with immediate answers to your product support questions and concerns.
- Update your StarTCC profile
- Make changes to your default E-mail address, phone number, and password whenever necessary.
- Display your site's licenses
- View a list of all the CA products for which your company site is currently licensed.
- Display StarTCC news items
- View and download recently published solutions for CA products, instructions for downloading from StarTCC, and helpful information for using CA-StarTrak, StarTCC, or other CA products.

Accessing the Technical Support Phone Services Directory

The Computer Associates Technical Support Phone Services Directory lists each CA product and the telephone number to call for primary support for that product. To access the Support Phone Services Directory, set your browser for http://supportconnect.ca.com and click Contact Us.

CA-TLC: Total License Care

Many CA software solutions use license keys or authorization codes to validate your hardware configuration. If you need assistance obtaining a license key or authorization code, contact the CA-TLC: Total License Care group through http://supportconnect.ca.com.

Contacting Technical Support

For further technical assistance with this product, please contact Computer Associates Technical Support at http://supportconnect.ca.com for a complete list of CA locations and phone numbers. Technical Support is available 24 hours a day, seven days a week.

If you are unable to resolve the problem, please have the following information ready before contacting Computer Associates Technical Support:

- All the diagnostic information described in Collecting Diagnostic Data.
- Product name, version number, operating system, and genlevel.
- Product name and version number of any other software you suspect is involved.
- Version level and PUTLEVEL of the operating system.
- Your name, telephone number, and extension (if any).
- Your company name.
- Your site ID.
- A severity code. This is a number (from one to four) that you assign to the problem.

  Use the following to determine the severity of the problem:

  1. A *system down* or inoperative condition.
  2. A suspected high-impact condition associated with the product.
  3. A question concerning product performance or an intermittent low-impact condition associated with the product.
  4. A question concerning general product utilization or implementation.

Generating a Problem Report

Once a Computer Associates Technical Support representative has determined that your problem requires further investigation, use the CAISERV utility to generate a problem report.

CAISERV Utility

The CAISERV diagnostic facility produces a problem report for you to fill out and send in with all problem documentation. CAISERV also produces a short report on the Computer Associates products that you have installed. You should also send this information to help Technical Support solve your problem.

To invoke CAISERV, execute the CAISERV proc in your sample JCL library:

```
// EXEC CAISERV,CAILIB='CAI.CAILIB',
// CAILPA='CAI.CAILPA',
// CAICICS='NULLFILE',
// SYSOUT=A
//
```

Edit the JCL to your installation's standards, and submit the job.

The messages you may encounter when running CAISERV are:

CAPP999E INSUFFICIENT STORAGE TO PROCESS CAISERV

Explanation: Sufficient storage was not allocated to execute CAISERV.

User Response: Use at least 100 KB of storage for executing CAISERV.

nonum ** PRODUCT CAISERV MODULE 'modulename' NOT ACCESSIBLE ***

Explanation: Libraries are not properly concatenated.

User Response: Review and modify the JCL. Execute CAISERV.

```
COMPUTER ASSOCIATES PROGRAM STATUS REPORT CAISERV- 1 PAGE  1
CAISERV 1.  89 2PP1  OPSYS='OS/39  ' dd mmm yyyy 9.24.23
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
> >
> COMPUTER ASSOCIATES PROBLEM REPORT FORM >
> >
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
> >
> CUSTOMER ID : _________ >
> COMPANY NAME : ___________________________________ >
> >
> COMPANY ADDRESS : ___________________________________ >
> ___________________________________ >
> ___________________________________ >
> >
> CONTACT NAME : _________________________ >
> >
> TELEPHONE NUMBER : _______________ EXTENSION : ____ >
> >
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
> >                                                                            >
> >                                                                            >
> HARDWARE INFORMATION                                                         >
> >                                                                            >
> CPU MODELS, STORAGE SIZES AND CPU IDS                                        >
> CPU 1 : ____ ____ ________________ CPU 2 : ____ ____ ________________        >
> CPU 3 : ____ ____ ________________ CPU 4 : ____ ____ ________________        >
> >                                                                            >
> DIRECT ACCESS STORAGE DEVICES                                                >
> DASD 1 : ________ DASD 2 : ________                                          >
> DASD 3 : ________ DASD 4 : ________                                          >
> >                                                                            >
> TAPE DEVICES                                                                 >
> TAPE 1 : ______ TAPE 2 : ______                                              >
> TAPE 3 : ______ TAPE 4 : ______                                              >
> >                                                                            >
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
> >                                                                            >
> SOFTWARE INFORMATION                                                         >
> >                                                                            >
> OPERATING SYSTEMS                                                            >
> OPSYS 1 : ________ OPSYS 2 : ________                                        >
> OPSYS 3 : ________ OPSYS 4 : ________                                        >
> >                                                                            >
> OTHER SYSTEM SOFTWARE                                                        >
> (VENDOR AND/OR IBM)                                                          >
> >     ______________________________________________________________________ >
> >
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
> >                                                                            >
> PROBLEM SEVERITY > PROBLEM TYPE                                              >
> >                                                                            >
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
> >                                                                            >
> >
> >
> >
> >
> >
> >
> >
> >
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
> >                                                                            >
```

Fill in the form completely. This information is logged into the Computer Associates Technical Support system and is tracked from the time it is reported to the time it is closed. On the next page of the problem report, describe the problem you are experiencing. The headings on this page look like this:

```
COMPUTER ASSOCIATES PROGRAM STATUS REPORT CAISERV- 1 PAGE  2
CAISERV 1.  89 2PP1  OPSYS='OS/39  ' dd mmm yyyy 9.24.23
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
> >
> COMPUTER ASSOCIATES PROBLEM REPORT FORM >
> >
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
> >
> PROBLEM DESCRIPTION: >
> >
> >
> >
> >
> >
> >
> >
> >
> >
> >
> >
> >
> >
> >
> >
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

The last page of the problem report provides a status of the Computer Associates products that you have installed. In most instances, this report includes the product release number, genlevel, and installation options. This is a sample report:

```
COMPUTER ASSOCIATES PROGRAM STATUS REPORT CAISERV- 1 PAGE 23
CAISERV 1.  95 9PP1  OPSYS='OS/39  ' dd mmm yyyy 9.47.21
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
> >
> CA-Product MVS >
> >
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CA-Product  : RELEASE R n. 6 GENLEVEL - 2002 5ASF   OS/39  LEVEL : SP6. .3 DFP :
3.3.21
CA-Product  BASE RELEASE - R 1. 6  m/dd/yy
SMF DUMP UTILITY RELEASE - R 1. 6  m/dd/yy
IEFU29 USER EXIT RELEASE - ??????
```

Product Versions and Maintenance

New users of Unicenter NetMaster Socket Management are provided with a distribution tape containing the current version of the system. Clients are requested to operate only under currently supported versions of Socket Management.

Clients with current maintenance agreements also receive ongoing Socket Management maintenance. When a new version of the system is available, a notice is sent to all current Socket Management clients.

Requesting Enhancements

Computer Associates welcomes your suggestions for product enhancements. All suggestions are considered and acknowledged. You can use either of two methods to request enhancements:

- Contact your Account Manager who will initiate a Demand Analysis Request (DAR) for you.
- Enter your request through StarTCC, the CA web-based, interactive support system at http://ca.com/support.

Chapter 9

Installation Verification Procedure (IVP)

This chapter describes the installation verification processes provided with the Unicenter Solve:CPT product. When you install and initially configure the product, the installation verification programs are also installed. With any service pack, release, or version change of the Unicenter SOLVE:CPT product you should always verify proper socket API (Application Programming Interface) workings of your installation and initial configuration in your environment. You can run the IVPs anytime you want to do a sanity check of your environment. Unicenter SOLVE:CPT provides two socket APIs, which are listed below. You should always verify both socket APIs.

If you followed the recommendations for using the distributed sample T09CONCP configuration member in the installation instructions in *Getting Started,* you should be ready to verify your installation. The T09CONCP configuration member is set up to run the IVPs for both the Unicenter SOLVE:CPT provided socket APIs.

- The Unicenter SOLVE:CPT API—The API at the heart of the high level sockets programming interface that supports the Unicenter SOLVE:CPT Tools themselves in addition to CICS application programmers.
- The EZASOKET/EZACICAL API—It is the API that is completely compatible with IBM's CICS sockets feature API.
- IVP for CPTMRO— This API is identical to the Unicenter SOLVE:CPT API IVP, with the exception that the listeners are remote to the CICS region.
- Verifying the Translation Process—Some guidelines on testing a translation table change

The Unicenter SOLVE:CPT API

This section describes the Installation Verification Procedure (IVP) for the Unicenter Solve:CPT API.

As noted previously:

- When you install and initially configure the product, the IVP is also installed.
- If you followed the recommendations for the distributed sample T09CONCP configuration member earlier in the *Getting Started* instructions, you should be ready to verify your installation of the Unicenter Solve:CPT API.

The T09CONCP configuration member contains definitions that allow a site to run the Unicenter SOLVE:CPT IVP over ports 1350 through 1353.

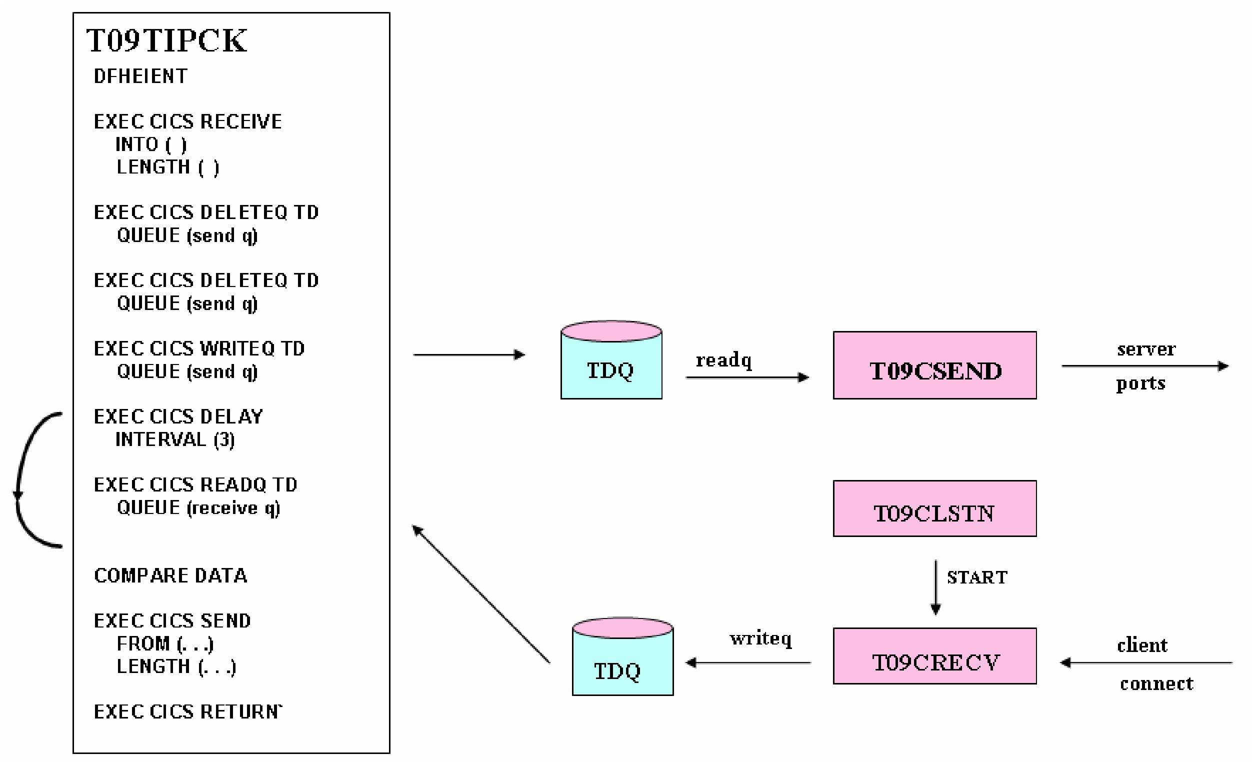
The following diagram is an excerpt from the T09CONCP configuration member sample.

It shows the correlation between:

- The Listen and Receive tools
- The client connection of the Send tool and how it connects to the server side Listen tool

```
.
.
.
         T09MCICS TRANSID=(IPPR,IPTL,IPST,IPT2),                X
               TRCSSN=ACTR,                                     X
               JOBNAME=TCPIP
*
*        Listen tool: Server definitions required
*                     for the IVP to run
*
         T09MLSTN PARM=IVPRECV1,                                X
               PORT=1350,                                       X
               TRANSID=IPTR,                                    X
               APISTAT=(CONN,TERM)
*
         T09MLSTN PARM=IVPRECV2,                                X
               PORT=1351,                                       X
               TRANSID=IPTR,                                    X
               APISTAT=(CONN,TERM)
.
.
.
*
*        Receive tool definitions required for the IVP to run
*
IVPRECV1     T09MRECV QNAME=IPRF,                               X
               OPTIONS=(RAW,FILE),                              X
               TRNSTAT=(TERM)
*
IVPRECV2     T09MRECV QNAME=IPRL,                               X
               OPTIONS=(LL),                                    X
               TRNSTAT=(TERM)
.
.
.
*
*        Send tool definitions required for the IVP to run
*
         T09MSEND QNAME=IPSF,                                   X
               PORT=1350,                                       X
               IPNAME=127.0.0.1,                                X
               OPTIONS=(RAW,FILE),                              X
               TRNSTAT=TERM,                                    X
               APISTAT=(CONN,TERM)
*
         T09MSEND QNAME=IPSL,                                   X
               PORT=1351,                                       X
               IPNAME=127.0.0.1,                                X
               OPTIONS=(LL),                                    X
               TRNSTAT=TERM,                                    X
               APISTAT=(CONN,TERM)
.
.
.
END
```

The following diagram shows the processing steps involved in the execution of the installation verification program.



The IVP is the CICS program T09TIPCK. This routine has a default transaction ID of IPCK. The program is initiated by an operator command and supports optional execution parameters.

This routine uses the Unicenter Solve:CPT send and receive tool transactions to test successful installation and availability of Unicenter Solve:CPT. This includes verifying the Unicenter Solve:CPT configuration definition statements and CICS resource definitions.

You can run the IVP anytime after Unicenter Solve:CPT is running in the CICS region.

Syntax

To run the IVP issue this CICS transaction ID:

IPCK *writeq readq*

Where:

writeq	The transient data queue name to which the IVP program will write, using the Unicenter Solve:CPT send tool. Default: IPSF.
readq	The transient data queue name from which the IVP program reads, using the Unicenter Solve:CPT receive tool. Default: IPRF.

Upon completion of the IVP transaction, a message appears on the system console and the CICS terminal indicating success or failure.

Note: This transaction can take up to 30 seconds to complete.

The following message is displayed upon successful completion of the IVP transaction:

```
+T09699I CPT IVP LOOPBACK TEST WAS SUCCESSFUL!!!
```

Also appearing in the CICS logging under CSML MSGUSR DD:

```
T09803I T01450 N CONNECTED TO LOCAL PORT 4096  FROM REMOTE PORT 1450, REMOTE HOST 127.0.0.1
T09807I API RECV STATS - APIRCV= 0      RCVBYT=0        RTNBYT=0        RCVTPL=0      LSTDATA=0
T09808I API SEND STATS - APISND= 1      SNDBYT=200      REQBYT=200      SNDTPL=0      LSTDATA=0
T09805I RECV (FILE) STATS - APIBYT=200      APIRCV=2      APIMAX=200    WTDBYT=200      WTD=1
WTDMAX=200
T09804I SEND (FILE) STATS - APIBYT=200      APISND=1      APIMAX=200    RTDBYT=200      RTD=1
RTDMAX=200
T09699I CPT IVP LOOPBACK TEST WAS SUCCESSFUL!!!
```

If the IVP transaction is unsuccessful, messages are written to the system console indicating a likely reason for the failure. Review the detailed description of the message and check the error log for additional information. If you do receive this message or get a failure message, you need to review your installation and initial configuration for completeness and accuracy to instructions, and call the CA support center for further guidance. The following table describes the four variations of the IVP.

Transaction Syntax 1	**Transaction Name 2**	**WriteQ 3**	**ReadQ 4**	**TCP Port Used 5**
IPCK IPSF IPRF	IPCK	IPSF	IPRF	1350
IPCK IPSL IPRL	IPCK	IPSL	IPRL	1351
IPCK IPSA IPRA	IPCK	IPSA	IPRA	1352
IPCK IPSS IPRS	IPCK	IPSS	IPRS	1353

1 The full syntax for each of the four IVP transactions for verifying basic product functionality.

2 The CICS transaction name of IPCK.

3 The name of the CICS TDQ to which the CPT send tool writes.

4 The name of the CICS TDQ to which the CPT receive tool writes.

5 Specifies which TCP/IP port is used by the IVP transaction for the test

Note: In column 1, there is a space between the transaction name, the writeq and the readq. Also, note that if you have changed the default names in the T09RDO member of T09SAMP library then the formats (names) listed above change accordingly.

To execute the IVP, type or cut and paste the syntax format from the table at any CICS terminal connected to a CICS region running Unicenter Solve:CPT.

Important! *You should run all four formats of the IVP.*

The EZASOKET/EZACICAL API

This section describes the Installation Verification Procedure (IVP) for Unicenter NetMaster Socket Management for CICS. Every Unicenter Solve:CPT customer will use this feature as an underlying API for Unicenter SOLVE:CPT API. The Unicenter Solve:CPT EZSOKET/EZACICAL API is completely compatible with the IBM CICS sockets API, which is also referred to as the CSKL type API or the EZASOKET API.

As noted previously:

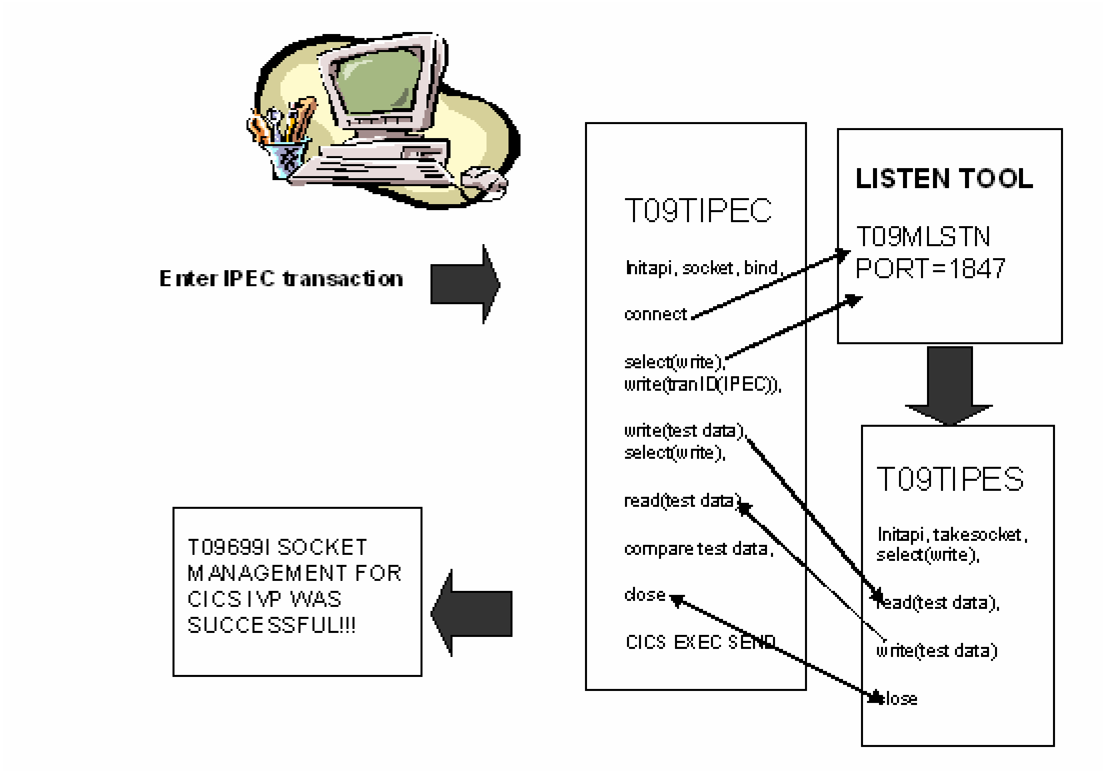
- When you installed and initially configured the product, the IVP is also installed.
- If you followed the recommendations for the distributed sample T09CONCP configuration member earlier in the *Getting Started* instructions, you should be ready to verify your installation of the Unicenter Solve:CPT EZSOKET/EZACICAL API.

The following is an excerpt from the recommendations for the distributed sample T09CONCP configuration member. Both the T09CONCP and T09CONEZ sample configuration members can run the EZASOKET IVP. The statements below are required to run the EZASOKET API IVP. The CSKL replacement server definition for port 1847 provides server functionality for the IVP:

```
T09MLSTN PORT=1847,                                         X
       SOCKCOMP=Y,       Required for CSKL type listeners   X
       CLNTRNS=NO,                                          X
       CLNTLEN=4,                                           X
       CLNTIME=1
```

If the product allows IBM to run its EZACIC01 TRUE exit then one could possibly specify the port as a parameter for a non-translate CSKL type listener during transaction initiation.

The following diagram shows how the EZASOKET API IVP processes the data packet.



The IVP is run using two CICS programs T09TIPEC and T09TIPES. The T09TIPEC routine has a default transaction ID of IPEC. The program is initiated by an operator command and supports optional execution parameters.

These routines use the EZASOKET API calls to send and receive TCP/IP data packets to test successful installation and availability of the EZASOKET API and Unicenter Solve:CPT. This includes verifying CPT configuration definition statements and CICS resource definitions.

You can run the IVP anytime after Unicenter Solve:CPT is running in the CICS region.

The IPEC transaction can be started with no parameters or it can optionally take a port parameter.

To run the IVP issue this CICS transaction ID:

IPEC ***portnumber***

Sample IPEC transactions:

```
IPEC
```

```
IPEC 1847
```

When the IPEC transaction works, the following message appears on the CICS terminal:

```
T09699I SOCKET MANAGEMENT FOR CICS IVP WAS SUCCESSFUL!!!
```

Should the IPEC IVP fail, it issues an error message to both the terminal and the CICS logs.

Sample IPEC error message displayed on the terminal:

```
T09224E T09TIPEC FAILED RTNCODE=0000000F DC=0000003D
```

Sample related error messages in the CICS logs for the above error:

```
00069 17:45:53 T09237E T09TIPEC CONNECT FAILED, RC=FFFFFFFF PROTOCOL ADDR=000204D28DCAC691
00069 17:45:53 T09233E T09TIPEC CALL CONNECT          ERRNO=0000003D  RC=FFFFFFFF
00069 17:45:53 T09224E T09TIPEC FAILED RTNCODE=0000000F DC=0000003D
```

IVP for CPTMRO

This section describes the IVP for Unicenter Solve:CPT CPTMRO feature.

As noted previously:

- When you install and initially configure the product, the IVP is also installed.
- If you followed the recommendations for the distributed samples T09CONMR and T09MRO00 configuration files, and the T09RDOMR member earlier in the *Getting Started* instructions, then you should be ready to verify your installation of the Unicenter SOLVE:CPT CPTMRO feature at this point.

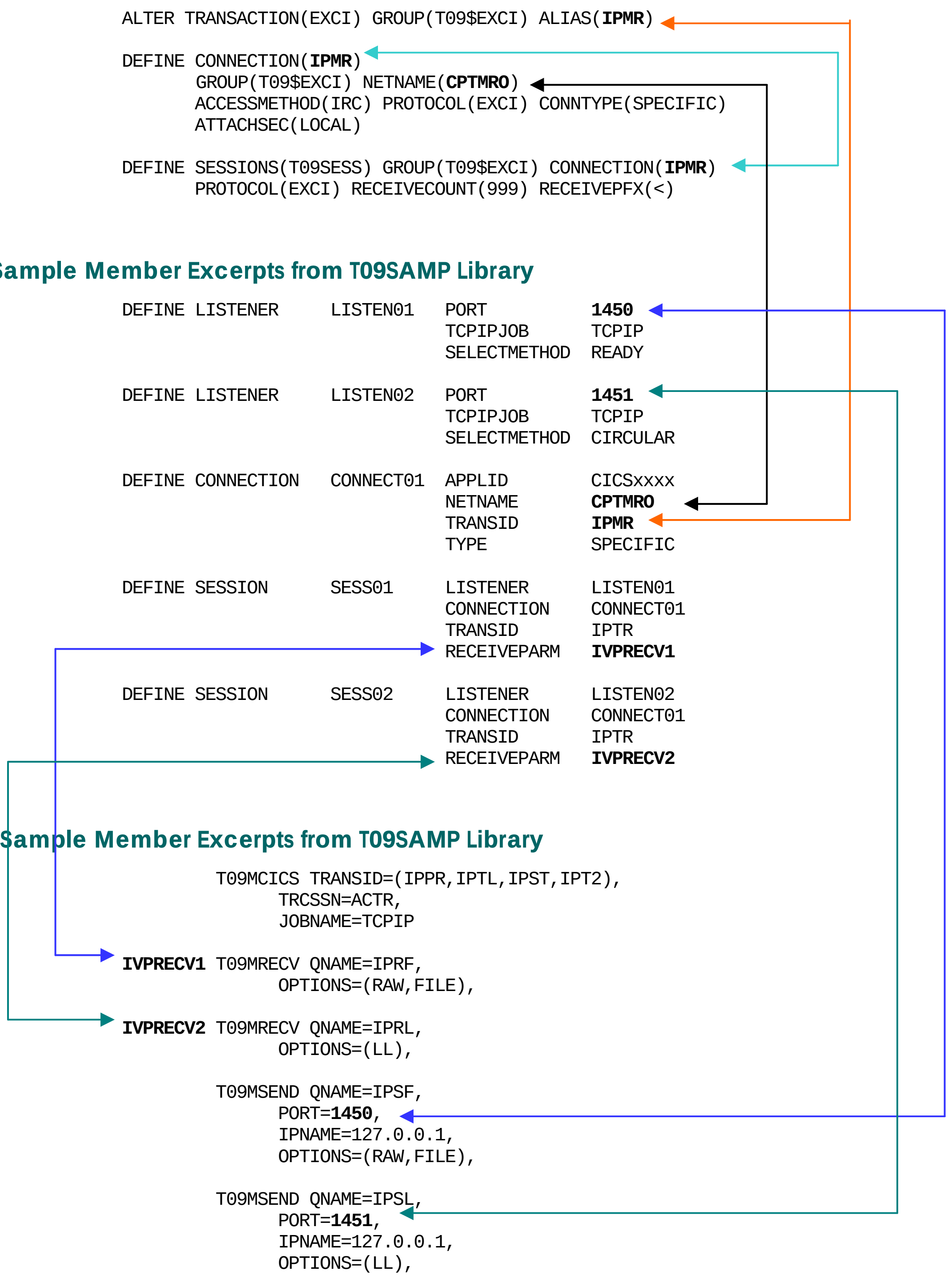
The configuration members contain definitions that allow a site to run the Unicenter SOLVE:CPT IVP over ports 1450 through 1453.

You run the IVP the exact same way as without CPTMRO, the only difference is that the listeners (servers) reside in a separate CPTMRO address space independent of the CICS region. Therefore, you will notice much of the documentation for the IVP for CPTMRO is almost identical to the Unicenter Solve:CPT API IVP.

This routine uses the Unicenter Solve:CPT send and receive tool transactions to test successful installation or Unicenter Solve:CPT availability. This includes Unicenter Solve:CPT and CPTMRO configuration definition statements, and CICS resource definitions.

The following diagram shows the correlations between all the configuration members within a CPTMRO environment, including that between the Listen and Receive tools. The sample also shows the correlation between the client connection of the Send tool and how it connects to the server side Listen tool.

These correlations are explained in detail in the sample configuration setup. See the “CPTMRO Installation and Configuration” chapter in *Getting Started*, and the “The CPTMRO Environment” chapter in this guide.



T09RDOMR Sample Member Excerpts from T09SAMP Library

```
ALTER TRANSACTION(EXCI) GROUP(T09$EXCI) ALIAS(IPMR)

DEFINE CONNECTION(IPMR)
       GROUP(T09$EXCI) NETNAME(CPTMRO)
       ACCESSMETHOD(IRC) PROTOCOL(EXCI) CONNTYPE(SPECIFIC)
       ATTACHSEC(LOCAL)

DEFINE SESSIONS(T09SESS) GROUP(T09$EXCI) CONNECTION(IPMR)
       PROTOCOL(EXCI) RECEIVECOUNT(999) RECEIVEPFX(<)
```

T09MRO00 Sample Member Excerpts from T09SAMP Library

```
DEFINE LISTENER     LISTEN01   PORT          1450
                               TCPIPJOB      TCPIP
                               SELECTMETHOD  READY

DEFINE LISTENER     LISTEN02   PORT          1451
                               TCPIPJOB      TCPIP
                               SELECTMETHOD  CIRCULAR

DEFINE CONNECTION   CONNECT01  APPLID        CICSxxxx
                               NETNAME       CPTMRO
                               TRANSID       IPMR
                               TYPE          SPECIFIC

DEFINE SESSION      SESS01     LISTENER      LISTEN01
                               CONNECTION    CONNECT01
                               TRANSID       IPTR
                               RECEIVEPARM   IVPRECV1

DEFINE SESSION      SESS02     LISTENER      LISTEN02
                               CONNECTION    CONNECT01
                               TRANSID       IPTR
                               RECEIVEPARM   IVPRECV2
```

T09CONMR Sample Member Excerpts from T09SAMP Library

```
         T09MCICS TRANSID=(IPPR,IPTL,IPST,IPT2),
               TRCSSN=ACTR,
               JOBNAME=TCPIP

IVPRECV1 T09MRECV QNAME=IPRF,
               OPTIONS=(RAW,FILE),

IVPRECV2 T09MRECV QNAME=IPRL,
               OPTIONS=(LL),

         T09MSEND QNAME=IPSF,
               PORT=1450,
               IPNAME=127.0.0.1,
               OPTIONS=(RAW,FILE),

         T09MSEND QNAME=IPSL,
               PORT=1451,
               IPNAME=127.0.0.1,
               OPTIONS=(LL),
```

You can run the IVP anytime after Unicenter Solve:CPT is running in the CICS region and CPTMRO is started.

You can run the CPT IVP to verify that the CPTMRO feature is operating properly. In order to do this, you must perform the following steps:

1. Start CPT in the CICS region—Below are the expected startup messages.

   In the following example, the IPST transaction is at a CICS terminal to start the Unicenter SOLVE:CPT product with the T09CONMR configuration file:

   ```
   IPST MR
   ```

   The following message appears on the CICS terminal:

```
T09181I T09TSTRT INITIALIZATION SUCCESSFUL FOR Unicenter SOLVE:CPT 6.1
```

   These messages display at the console during CPT initialization:

```
+T09180I T09TSTRT STARTING Unicenter SOLVE:CPT 6.1
+T09183I T09TSTRT LMP Code=ZD,   STARTRAK=SCPT       Abbreviation is Unicenter TCPaccess
+T09100I T09TSTRT CPT TRUE EXIT INTERFACE ENABLED
+T09111I T09CINIT DEFAULT TRANSLATION TABLE T09XENG  LOADED
+T09181I T09TSTRT INITIALIZATION SUCCESSFUL FOR Unicenter SOLVE:CPT 6.1
```

2. Start the CPTMRO procedure.

Important! *To properly initialize, CPT must first be started in the CICS region before starting the external CPTMRO address space.*

Typically, you should see something similar to the subset below:

```
T09M0000I CPT/MRO Version 6.1 initializing....
T09M0038I Log task started.
T09M0012I Operator interface enabled.
T09M0023I CPT/MRO Version 6.1 now available.
T09M0028I Processing startup Exec 'T09MRO00'....
T09M0030I Startup Exec processing complete.
T09L0003I Logging now active.
T09M0000I CPT/MRO Version 6.1 initializing....
T09M0038I Log task started.
T09M0012I Operator interface enabled.
T09M0023I CPT/MRO Version 6.1 now available.
T09M0028I Processing startup Exec 'T09MRO00'....
T09M0030I Startup Exec processing complete.
T09L0003I Logging now active.c
T09IL001D Listener 'LISTEN01' general initialization successful
T09IC014D Connection 'CONNECT01' Connection manager EXCI request successful.
T09IC001D Connection 'CONNECT01' general initialization successful
T09IC003D Connection 'CONNECT01' processing begins...
T09MX035I Connection 'CONNECT01' startup complete
T09IL011D Listener 'LISTEN01' listening for connections on port 1450
T09IL003D Listener 'LISTEN01' processing begins...
T09MX035I Listener 'LISTEN01' startup complete
T09MX035I Session 'SESS01' startup complete
T09IC014D Connection 'CONNECT01' Connection manager EXCI request successful.
```

3. To run the IVP—type in **IPCK** at a CICS terminal connected to the CICS running CPT configured for CPTMRO. You should get the following response at the terminal within 30 seconds:

```
T09699I CPT IVP LOOPBACK TEST WAS SUCCESSFUL!!!
```

In the CPTMRO address space logging:

```
T09IC014D Connection 'CONNECT01' Connection manager EXCI request successful.
T09IL035D Listener 'LISTEN01' end point passed to CICSxxxx on CONNECT01
T09IL036D Listener 'LISTEN01' end point accepted by CICSxxxx on CONNECT01
T09IC014D Connection 'CONNECT01' Connection manager EXCI request successful.
T09IL029D Listener 'LISTEN01' Connection EXCI call successful for CONNECT01.
```

In the CICS logging under CSML MSGUSR DD:

```
T09803I T01450 N CONNECTED TO LOCAL PORT 4096  FROM REMOTE PORT 1450, REMOTE HOST 127.0.0.1
T09807I API RECV STATS - APIRCV= 0      RCVBYT=0        RTNBYT=0        RCVTPL=0      LSTDATA=0
T09808I API SEND STATS - APISND= 1      SNDBYT=200      REQBYT=200      SNDTPL=0      LSTDATA=0
T09805I RECV (FILE) STATS - APIBYT=200      APIRCV=2      APIMAX=200    WTDBYT=200      WTD=1
WTDMAX=200
T09804I SEND (FILE) STATS - APIBYT=200      APISND=1      APIMAX=200    RTDBYT=200      RTD=1
RTDMAX=200
T09699I CPT IVP LOOPBACK TEST WAS SUCCESSFUL!!!
```

Similar to CPT API IVP documented above, here is the full syntax for each of the four IVP transactions for verifying the CPTMRO interface:

The IVP is CICS program T09TIPCK. This routine has a default transaction ID of IPCK. The program is initiated by an operator command and supports optional execution parameters.

Syntax

To run the IVP issue this CICS transaction ID:

IPCK *writeq readq*

writeq	The transient data queue name to which the IVP program write, using the Unicenter Solve:CPT send tool. Default: IPSF
readq	The transient data queue name from which the IVP program reads, using the Unicenter Solve:CPT receive tool. Default: IPRF

Upon completion of the IVP transaction, a message appears on the system console and CICS terminal indicating success or failure.

Note: This transaction can take up to 30 seconds to complete.

The following message we be displayed on the CICS terminal upon successful completion of the IVP transaction:

```
+T09699I CPT IVP LOOPBACK TEST WAS SUCCESSFUL!!!
```

If the IVP transaction is unsuccessful, a message written to the system console indicating a likely reason for the failure. Review the detailed description of the message and check the error log for additional information. If you do receive this message, or get a failure message you need to review your installation and initial configuration for completeness and accuracy to instructions, and call the CA support center for further guidance.

The following table below describes the four variations of the IVP.

1. The full syntax for each of the four IVP transactions for verifying basic product functionality.
2. Separates out the CICS transaction name of IPCK.
3. The name of the CICS TDQ to which the CPT send tool writes.
4. The name of the CICS TDQ to which the CPT receive tool writes.
5. Specifies which TCP/IP port is used by the IVP transaction for the test

Transaction Syntax 1	Transaction Name 2	WriteQ 3	ReadQ 4	TCP Port Used 5
IPCK IPSF IPRF	IPCK	IPSF	IPRF	1350
IPCK IPSL IPRL	IPCK	IPSL	IPRL	1351
IPCK IPSA IPRA	IPCK	IPSA	IPRA	1352
IPCK IPSS IPRS	IPCK	IPSS	IPRS	1353

Note: Column1 has a space between the transaction name, the writeq and the readq. Also, if you have changed the default names in the T09RDO member of T09SAMP library then the formats listed above change accordingly.

To execute the IVP, simply type or cut and paste from the above table the syntax format at any CICS terminal connected to a CICS region running Unicenter Solve:CPT.

Important! *You should run all four formats of the IVP.*

Verifying the Translation Process

To verify that the translation process is working properly, set APITRAC=CLTD or the ACM tracing option, ACMTCLTD on. Either of these settings causes data to be traced with messages T09931I and T09933I.

Use the UNIX client tool (cl1), delivered with the Unicenter SOLVE:CPT product, to send ASCII input to a Client/Data Listener. See appendix “ UNIX Test Programs” for more information.

Syntax

```
cl1 -d -o all -f ibmfmt -p nnnn host
```

Where:

ibmfmt	File containing a minimum of a one- to four-character transaction ID defined in your CICS.
nnnn	Port number of the Unicenter SOLVE:CPT Listener Service.
Host	Host name or IP address.

Chapter
10

Translation Tables

This chapter describes the translation tables available for Unicenter Solve:CPT. It includes these sections:

- Choosing a Unicenter SOLVE:CPT Translation Table—Details factors to consider when selecting a Unicenter SOLVE:CPT translation table
- National-Use Characters—Describes the National-Use characters
- Modifying or Adding Translate Tables—Describes how to add new translate table or edit the existing tables
- Translate Table Modules Structure—Details information about the structure of translate tables provided with Solve: CPT
- Generating Prefixes for National Language Translate Tables—Describes the use of the XLTBL macro
- Customizing the Translation Table—Describes how to install customized translation tables.

Choosing a Unicenter SOLVE:CPT Translation Table

Because of translation differences between languages, when using a table other than English for Unicenter SOLVE:CPT translation, you should first do the following:

1. Determine if any of the characters indicated in the following table appear in your Unicenter SOLVE:CPT input or output stream.
2. If there are any, check the translate table source to verify they translate correctly.

The SMTP service uses the @ character as part of the mailing address. In the Danish table however, an EBCDIC X’7C’ does not represent an @. Rather, in that table an ASCII X’40’ is translated to an EBCDIC X’40’ (which is a blank), and an EBCDIC X’7C’ is translated to an ASCII X’5C’. Therefore, the Danish table cannot be used for SMTP.

National-Use Characters

The following table indicates the 14 national-use characters.

- The first row is the EBCDIC character representation for English
- The second row is the EBCDIC hexadecimal equivalent
- The third row is the seven-bit ASCII hexadecimal equivalent

Note: The translation of these characters may vary from language to language.

CHAR	**¢**	**!**		**‘**	**$**	**#**	**@**	**¬**	**~**	**{**	**}**	\	**|**	**“**
HEX(E)	4A	5A	6A	79	5B	7B	7C	5F	A1	C0	D0	E0	4F	7F
HEX(A)	no symbol	21	7C	60	24	23	40	5E	7E	7B	7D	5C	ns	22

This information was taken from a table in the *IBM 3270 Information Display System Character Set Reference, GA27-2837.*

Modifying or Adding Translate Tables

If necessary, you can modify the translate tables to suit customer requirements. In addition, other translate tables with similar structure can be added.

Tip: It is recommended that you develop new tables using an existing table as a model.

Translate Table Modules Structure

A translate table module has a special structure containing a set of translate tables as described below. The national language translate table modules used by the Unicenter SOLVE:CPT and Unicenter SOLVE:CPT-based applications must have this structure:

```
Prefix - XLTBL TYPE=START

Set of translate tables
Suffix - XLTBL TYPE=FINISH
```

For Unicenter SOLVE:CPTpurposes, there should be four translate tables in the set.

- An ATOE table for ASCII-to-EBCDIC translation
- ETOA for EBCDIC-to-ASCII translation
- AUPC for ASCII lower case to upper case folding
- EUPC for EBCDIC lower case to upper case folding

Each individual table is 256 bytes long and suitable for use with the IBM System/370 TR instruction. Additionally, the statement labels for these tables must be listed in the XLTBL macro TABLE parameter in the order listed above.

Data translation requires only the first two tables (ATOE and ETOA), as there are no case changes involved in the transfer. Various command line entries require all four translate tables in the set.

Generating Prefixes for National Language Translate Tables

The XLTBL macro generates the prefix and suffix needed by a national language translate table module.

XLTBL Macro Syntax

The syntax for the XLTBL macro is as follows:

```
[ name ] XLTBL [ TYPE = ( START | FINISH ) ]
        [, KIND = ( CHAR | NUMBS | DBCS) ]
        [, TABLE = label_name, ., ., . ]
        [, FLAG = ( REFLEX | ASCII8 | ASCI16 | NOSOSI | BKTSUB ) ]
        [, TITLE = 'char_string' ]
        [, TAG = character ]
        [, ID = character ( s ) ]
        [, DSECT = ( YES | NO ) ]
```

TYPE = (START | FINISH)

Specifies whether header or trailer data should be generated.

START generates the translate module prefix, FINISH its suffix.

Default: START.

KIND = (CHAR | NUMS | DBCS)

This keyword defines the translation set you are defining:

- CHAR: single byte, four-table set
- NUMS: single byte, two-table set

Default: CHAR.

TABLE = (*label_name*, *label2*, *label3*, ...)

List of addresses for individual translate tables (*label1*, *label2*, ...)] contained in the translate table module.

label1	The statement label for the first translate table
label2	The second, and so forth

The number of items depends upon KIND.

For Solve: CPT translate tables, KIND=CHAR should be coded, and TABLE should list four labels for ATOE, ETOA, AUPC, and EUPC translate tables, in that order.

Default: None.

FLAG = (REFLEX | ASCII8 | ASCI16)

List of options to set in the flag area of the fixed header:

REFLEX	ASCII to EBCDIC and EBCDIC to ASCII is symmetric.
ASCII8	ASCII data is eight bit (SBCS)
ASCI16	ASCII data is 16 bit

Default: None.

TITLE = '*char_string*'	Specifies a character string used to describe the translation set. Length is limited to 30 characters.

The default is '*name* Translation', where name is the label on the XLTBL invocation statement.

TAG = *character*	One-byte character identifier used to group the soft key definitions.

Default: $

ID = *character*(s)	Three-byte character identifier used as a root for macro generated label names.

Note: Only change this identifier if the XLTBL macro is issued multiple times in the same TYPE.

Default: XS@

DSECT = YES | NO

YES	Generates a DSECT map of the data area.
NO	Generates a CSECT map of the data area.

Note: When you create your own translation tables, the DSECT parameter should always be set to NO. Although the default value of YES is used by Unicenter Solve:CPT for internal applications, if YES is selected when you create a translation table, the translation table will be invalid.

When YES is specified the keywords TITLE, FLAG, and TABLE are ignored, so this parameter should **always** be set to NO.

Default: YES.

XLTBL Macro Example

```
ENGLISH XLTBL TYPE=START,
KIND=CHAR,
TABLE=(ATOE,ETOA,AUPE,EUPA),
FLAG=(ASCII8),
TITLE='English Single byte table',
DSECT=NO
```

Maintaining Translation Tables Using UMODTRAN

The SMP/E usermod JCL stream UMODTRAN in the SAMP library provides for translation table maintenance. It is used for both modifying and adding tables. It compiles and links the table source into a load module in the LOAD library. Unicenter Solve: CPT uses only the load module version when it does translation.

SMP/E maintains all translate table source in the SAMP library. For existing tables, make changes in the SAMP member before executing UMODTRAN. Build new translate tables in a different library, as UMODTRAN copies them into the SAMP library as part of its SMP/E processing. Once the table is copied, all subsequent changes should be made in the SAMP library.

Note:

1. You should run an SMP/E APPLY CHECK against any USERMOD that you are trying to install, as there may be additional PREs on your system that are not accounted for.
2. Once you gather this information, add the SYSMOD list(s) to the ++PRE(*xxxxxx*) statement.
3. SMP/E REJECT the USERMOD to remove the invalid entry from the SMP/E CSI.
4. You can now RECEIVE/APPLY the USERMOD with success.

Directions for using UMODTRAN are located within the member. When needed, make a copy of the UMODTRAN provided and adapt it to your environment.

Customizing the Translation Table

The ability to modify or create new translation tables is controlled through the SMP/E USERMOD facility. To incorporate a modified or new translation table into Unicenter SOLVE:CPT, you need to receive and apply the modules to SMP/E CSI as described below.

It is recommended that sites do not modify the default or distributed translation table named T09XENG. Distributed maintenance could regress site modifications.

A default translation table is specified in the T09MCICS configuration macro. This default is used for all translation functions unless overridden. You can override the default by another configuration macro statement at a lower level. For example you can override in the T09MLSTN macro or in the CPT/API translate call.

Customization Steps for Changing Translation Tables

To use an alternate translation table, do the following:

1. Create a new member name and copy in the table from sample member T09XENG in your T09SAMP library or any sample member in the SAMP library..

   Note:

   - Do not use the T09XENG or any other member directly since it may be overridden by future SMP/E maintenance
   - The member must be in the T09SAMP library for Step 3 usermod to work as shown in the following example.

2. Customize the table created in Step 1 to fit your needs.
3. Create a valid program table entry in an installed Resource Definition Online (RDO) group using the newly created name from Step 1.

   An example of this is:

```
DEFINE PROG (T09XENG)  LANG (ASSEMBLER) DA (ANY) GROUP (T09CPT)
```

4. Create a new JCL member name and copy in sample member T09UTRAN from the T09SAMP library. A sample of this JCL is shown below.

   Make the following updates before submitting the job to apply a new translation table:

   a. Copy in a valid JOBCARD.
   b. Globally change SMPINDX to your SMPE high level qualifier.
   c. Change " ++ SRC (T09XENG) " to the name of the table you created.
   d. Verify FMID(C2K6100), this is valid for Unicenter SOLVE:CPT 6.1.

```
//<NAME>  JOB (000), 'TRANS USER MOD'
//*
//SMPE     EXEC PGM=GIMSMP,REGION=4096K,TIME=960,
//         PARM='CSI=SMPINDX.CSI,PROCESS=WAIT'
//SMPHOLD  DD  DUMMY
//SMPLOG   DD  DSN=SMPINDX.SMPLOG,DISP=MOD
//SMPOUT   DD  SYSOUT=*
//SMPPTFIN DD  *
++ USERMOD (MU1TRAN) .
++ VER (Z038)
   FMID(C2K6100)
++ SRC (T09XENG) TXLIB(T09SAMP) DISTMOD(T09LOAD) DISTLIB(AT09SAMP)
/*
//SMPCNTL  DD  *
 SET BDY(GLOBAL) .
 RECEIVE S(MU1TRAN) .
 SET BDY(TCPTZN) .
 APPLY   S(MU1TRAN) .
/*
```

5. Verify that the table is contained in a library that is part of the CICS/TS DFHRPL concatenation.
6. Restart CICS/TS if changes were needed in Step 4.

Restart Unicenter SOLVE:CPT.

Appendix

A Security

This appendix provides information on the security features for Unicenter SOLVE:CPT.

It includes the following topics:

- About the Optional Security Program—Describes the connection Unicenter SOLVE:CPT
- Transaction Security— Describes how to secure CPT transactions to prevent unauthorized use
- Unicenter NetMaster Command Processor Security—Describes how to secure the remote commands coming into the command processor from the Unicenter NetMaster for TCP/IP interface
- CPTMRO Security—In addition to the Security Program, with CPTMRO you have an easy to configure firewall PDS member

About the Optional Security Program

This section describes how to invoke and use Unicenter SOLVE:CPT 's optional security program feature provided with the LISTEN service.

Security is provided via a program for user evaluation of requests via IP address or user ID/password for the services of local listeners/servers. If a security program is implemented (configured), the user program is invoked for each connection request. If desired, the user program can be specified for each listener. The appropriate server transaction is initiated if authorized by the user security program. Otherwise, the client is notified that the connection is terminated.

To implement the security program, the SCTYEXIT=*program-name* must be coded in the T09MCICS macro or the T09MLSTN macro of the T09CONcp configuration table. This user program is EXEC CICS LINKed during the connection process, must conform to CICS coding standards, and be defined as a RDO program entry.

Note:

- If no SCTYEXIT parameter is coded in the configuration table, all connection requests are authorized and the user ID will be the same as the Listener transaction
- If SCTYEXIT is coded but the program is missing or is disabled, no connections are allowed

Each Listen tool can specify its own security program. If the Listen tool does not specify a security program and the SCTYEXIT parameter is coded on the T09MCICS configuration macro, then that program is used as the security program.

In other client/server designs, the application receives control when the connection is made and should make any desired security checks before starting server activity.

Note: However, by coding SCTYTYPE=MANDTORY and SCTYEXIT=*program-name* on the T09MCICS configuration macro, the security program name that is coded will always be executed regardless of what is coded at the T09MLSTN(listen tool) level or at the user-written listener before returning control to the application.

Security Program Logic Flow

The user security program is responsible not only for making the determination of whether a connection is authorized, but also for any desired logging or other capture of unauthorized requests. Because the program will be driven for each connection on a listener, performance implications should be considered in designing security programs.

When security is specified in the configuration table, a new transaction is started (the program is T09TLST2 with transaction ID IPT2). This transaction then CICS link to the specified security program. The program is passed the Security Communications Block (SCB). It contains fields used to determine the validity of the connection. One of the fields in the SCB is the token or socket ID of the connection. The token can be used to initiate SEND and RECEIVE calls in order to communicate with the remote client to determine a user ID, password, or any other identifying characteristics. Any of the other fields in the SCB can be used.

Security Program Return

Upon return from the security program, four SCB fields are used:

- The authorization switch is used to authorize the connection by setting a character one in the field.
- The terminal facility is used to specify a CICS termid to associate with the new CICS/TS transaction to be EXEC CICS STARTed.
- If the user ID field is specified the new transaction is STARTed with that user ID.
- The transaction to be started can also be modified by the security program. If changed then the newly specified transaction is STARTed.

Note: When termid and user ID are both specified, user ID takes precedence.

The security program can perform additional SEND and RECEIVE calls to request and retrieve data. This data might be some form of user ID or password. The program can then verify the user ID and password with the EXEC CICS VERIFY command. If the user ID is returned in the SCB, the new transaction is started with EXEC CICS START USERID (user ID).

The Security Communications Block

The connection process transaction and the user security program communicate through the SCB. Unicenter SOLVE:CPT provides information about the request and its origin. The user security program determines whether the request is authorized and, optionally the name of a terminal facility or user ID to associate with an EXEC CICS STARTed server transaction. A DSECT of the SCB for assembler programs can be generated with the T09DSCTY macro.

This is what the T09DSCTY DSECT control block looks like in assembler language:

```
Name Operation Operands Description

SECPARM  DSECT
SECTRAN  DS  CL4   SERVER TRANSACTION REQUESTED
SECDATA  DS  XL40  REQUESTOR DATA
SECSTRT  DS  CL2   HOW TASK IS TO BE STARTED
SECICTM  DS  XL6   INTERVAL CONTROL TIME
SECADRS  DS  0CL8  REQUESTOR ADDRESS
SECAFAM  DS  H     DOMAIN
SECRPRT  DS  H     PORT
SECRHST  DS  F     HOST IP ADDRESS
SECACTN  DS  CL1   PERMIT/PROHIBIT SWITCH
SECPRMT  EQU C’1’  ..OKAY, INITIATE TASK
         DS  X     RESERVED
SECTMID  DS  CL4   ANY ASSOCIATED CICS TERMINAL
SECLPRT  DS  H     LOCAL SERVER PORT
SECUSER  DS  CL8   USER ID
         DS  CL512 RESERVED
SECTOKN  DS  F     TOKEN - ENDPOINT
SECLHST  DS  F     LOCAL HOST
```

```
*
SECLEN   EQU *-&LABEL LENGTH OF SECURITY DATA AREA
```

The following table describes the data contained in the DSECT.

Field	Format	Description
SECTRAN	4-byte character	Requested server transaction. May be modified by the program.
SECDATA	40-byte character	Client data, if available.
SECSTRT	2-byte character	Method of server initiation: KC, TC, or IC.
SECICTM	6-byte character	IC Hours, Minutes, Seconds.
SECAFAM	2-byte binary	Address family: Inet domain=2.
SECRPRT	2-byte binary	Client remote port number.
SECRHST	4-byte binary	Client remote host IP address.
SECACTN	1-byte character	Authorization switch: 1=accept 0=fail
SECTMID	4-byte character	Associated terminal facility.
SECLPRT	2-byte binary	Requested server local port.
SECUSER	8-byte binary	Returned user ID.
SECTOKN	4-byte binary	Token that represents the TCP connection.
SECLHST	4-byte binary	Local host IP address.

Transaction Security

Access to Unicenter SOLVE:CPT can be limited by using normal CICS/TS transaction security. Regardless of whether an external security manager or native CICS/TS security is used, the following CICS/TS transactions should be protected:

- IPST—Unicenter SOLVE:CPT Interface Initialization
- IPPR—Unicenter SOLVE:CPT Interface Termination
- IPAO—Administrator Interface Operations Control

For further granularity in transaction security the following tables have been provided for your convenience.

In these tables, the IPA* transactions are into two groups.

- One Group of transactions only display the CICS CPT environment
- The second group of transactions alter the CICS CPT environment

The following table lists the IPA* transactions that **DISPLAY** the CICS CPT environment:

T09ABDTL	IPAB
T09AHELP	IPAK
T09ACNFG	IPAC
T09AMAIN	IPAM
T09ATLST	IPAL
T09AMENU	IPAN
T09APING	IPAP
T09ATSND	IPAS
T09AUTIL	IPAU
T09AWTCH	IPAW

The following table lists the IPA* transactions that **ALTER** the CICS CPT environment:

T09ATADD	IPAA	Add/Alter Tool
T09AQCLS	IPAE	Close Connections
T09AGENT	IPAG	Generate Trace
T09AOLWT	IPAI	Online Queue Write
T09AOCTL	IPAO	Operations Menu
T09AQCLS	IPAQ	Close Connections
T09ARSTQ	IPAR	Reset Trace/Error Queues
T09ATROP	IPAT	Add/Alter Trace Options

T09AYANK	IPAY	Online Queue Release
T09AZAPS	IPAZ	Reset Statistics
T09TTERM	IPPR	CPT Interface Termination
T09TSTRT	IPST	CPT Interface Initialization

Note: If you changed the AIPREFIX=IPA parameter on the T09MCICS macro in the Unicenter SOLVE:CPT configuration file, that the transaction names above will have changed.

Unicenter NetMaster Command Processor Security

Unicenter NetMaster command processor server security is activated via T09MCMDS configuration and, by default, is fully enabled. The T09MCMDS configuration is described in the chapter "Configuration Reference." Security is optional. However, it is highly recommended that security **not** be disabled to ensure that unauthorized users are denied access.

Unicenter NetMaster command processor server security ensures that users have valid user IDs and passwords, and that users have the correct authority level to enter command server requests.

When the command processor security is enabled (by specifying SECURITY=Y on the T09MCMDS macro in the T09CON*xx* configuration file), all users requesting a connection with the command server must have:

- A valid user ID and password registered with the ESM (external security manager) under which the Unicenter NetMaster command processor is running.
- An entry in the ESM's general resource class profile (for example, FACILITY) must be defined so the command processor can verify that even valid users have the proper authority to enter command server requests.

The facility resource name can be up to 44 characters and must be configured in the ESM's general resource profile (FACILITY) and on the SECENT parameter of the T09MCMDS macro in the T09CON*xx* configuration file.

There are two levels of FACILITY command authority access:

- READ access: Allows users to issue commands to display session, server and global statistics.
- UPDATE access: Allows users to issue commands to alter (and display) the CICS environment, and:
  - Stop and start servers

- Terminate sessions
- Shutdown and restart the product
- Initiate CICS transactions

Security Configuration for eTrust CA-ACF2

Many sites have both production and test CICS regions. You should give operators and system personnel the proper authority to do their job. At the same time, you want to protect the CICS applications from overly inquisitive or unruly personnel in the user community.

The 44 characters resource name must match the SECENT parameter of the T09MCMDS macro in the T09CON*xx* configuration file. Many sites have a requirement that personnel have authority levels appropriate for both production and development CICS regions.

Once you define these entities in the FACILITY resource class, you must then give personnel the proper authority to do their job.

The default access of NONE prevents users from using the Unicenter NetMaster command/control server.

READ access allows users to display session, server and global statistics.

UPDATE access enables users to start and stop transactions, sessions, servers and applications.

The following are sample FACILIY entity definitions for use by MVS system personnel (SYS1), where:

CICSOPR	A CICS operator
CICSDEV	A CICS developer
$SKTVIEW.CICSPROD.CMDAUTH	Configured in the SECENT parameter for the CICS production regions
$SKTVIEW.CICSTEST.CMDAUTH	Configured in the SECENT parameter for the CICS development regions

eTrust CA-ACF2 Example

```
SET R(FAC)
COMP
$KEY($SKTVIEW.CICSPROD.CMDAUTH) TYPE(FAC)
UID(SYS1-)          SERVICE(READ,UPDATE)
UID(CICSOPR-)       SERVICE(READ,UPDATE)
UID(CICSDEV-)       SERVICE(READ)
STORE
COMP
$KEY($SKTVIEW.CICSTEST.CMDAUTH) TYPE(FAC)
UID(SYS1-)          SERVICE(READ,UPDATE)
UID(CICSOPR-)       SERVICE(READ,UPDATE)
UID(CICSDEV-)       SERVICE(READ,UPDATE)
STORE
```

Security Configuration for eTrust CA-Top Secret

Many sites have both production and test CICS regions. You should give operators and system personnel the proper authority to do their job. At the same time, you want to protect the CICS applications from overly inquisitive or unruly personnel in the user community.

The 44 characters resource name must match the SECENT parameter of the T09MCMDS macro in the T09CON*xx* configuration file. However, eTrust CA-Top Secret mandates that sites only use up to the first eight characters in the IBMFAC resource class. Therefore, you could define the IBMFAC entry as follows:

```
TSS ADD(SYSADM) IBMFAC($SKTVIEW)
```

Example 1

Many sites have a requirement that personnel have authority levels appropriate for both production and development CICS regions. The PERMIT command allows one to place the full 44 -character entity name for an IBMFAC entry.

The default access of NONE prevents users from using the Unicenter NetMaster command/control server.

READ access enables users to display session, server and global statistics.

UPDATE access allows users to start and stop transactions, sessions, servers and applications.

Once the entities are defined in the FACILITY resource class, you must give personnel the proper authority to do their job.

The following are sample FACILIY entity definitions for use by MVS system personnel (SYS1), where:

CICSOPR	CICS operator.

CICSDEV	A CICS developer.

$SKTVIEW.CICSPROD.CMDAUTH
Configured in the SECENT parameter for the CICS production regions.

$SKTVIEW.CICSTEST.CMDAUTH
Configured in the SECENT parameter for the CICS development regions.

Example 2

```
TSS PERMIT(SYS1,CICSOPR)  IBMFAC($SKTVIEW.CICSPROD.CMDAUTH) ACCESS(UPDATE)
TSS PERMIT(SYS1,CICSOPR)  IBMFAC($SKTVIEW.CICSTEST.CMDAUTH) ACCESS(UPDATE)
TSS PERMIT(CICSDEV)       IBMFAC($SKTVIEW.CICSPROD.CMDAUTH) ACCESS(READ)
TSS PERMIT(CICSDEV)       IBMFAC($SKTVIEW.CICSTEST.CMDAUTH) ACCESS(UPDATE)
```

Required TSS APARs

For command authority security checking to work properly, you must:

- Install eTrust CA-Top Secret APAR KEC5385.
- You must also specify "OPTIONS(70)" in the TSS startup parameter file and recycle TSS. KEC5385 allows the CICS principal facility (terminal) associated with the command server to be defined as an output-only device (see topic "Define the principal facility to be associated with T09TCMDS").

Note: The "OPTIONS(70)" parameter causes KEC5385 to be dynamically implemented. Without this APAR, command server authorization is essentially bypassed, and full command authority will be granted to all users with valid user IDs and passwords that sign on to the command server.
Contact TSS technical support for details and assistance in obtaining and installing KEC5385.

Security Configuration for RACF

Many sites have both production and test CICS regions. You should give operators and system personnel the proper authority to do their job. At the same time, you want to protect the CICS applications from overly inquisitive or unruly personnel in the user community.

The 44 characters resource name must match the SECENT parameter of the T09MCMDS macro in the T09CON*xx* configuration file. Many sites have a requirement that personnel have authority levels appropriate for both production and development CICS regions.

Here are two sample FACILIY entity definitions for use by a production and a CICS region:

Example 1

```
RDEFINE FACILITY $SKTVIEW.CICSPROD.CMDAUTH UACC(NONE)
RDEFINE FACILITY $SKTVIEW.CICSTEST.CMDAUTH UACC(NONE)
```

The default access, NONE, prevents users from utilizing the Unicenter NetMaster command/control server.

READ access enables users to display session, server and global statistics.

UPDATE access enables users to start and stop transactions, sessions, servers and applications.

Once the entities are defined in the FACILITY resource class, you want to give personnel the proper authority to do their job.

The following are sample FACILIY entity definitions for MVS (SYS1) system personnel to use, where:

CICSOPR	A CICS operator.
CICSDEV	A CICS developer.

$SKTVIEW.CICSPROD.CMDAUTH
Configured in the SECENT parameter for the CICS production regions.

$SKTVIEW.CICSTEST.CMDAUTH
Configured in the SECENT parameter for the CICS development regions.

Example 2

```
PERMIT $SKTVIEW.CICSPROD.CMDAUTH CLASS(FACILITY) ACCESS(UPDATE) ID(SYS1,CICSOPR)
PERMIT $SKTVIEW.CICSPROD.CMDAUTH CLASS(FACILITY) ACCESS(READ)   ID(CICSDEV)
PERMIT $SKTVIEW.CICSTEST.CMDAUTH CLASS(FACILITY) ACCESS(UPDATE) ID(SYS1,CICSOPR)
PERMIT $SKTVIEW.CICSTEST.CMDAUTH CLASS(FACILITY) ACCESS(UPDATE) ID(CICSDEV)
```

CPTMRO Security

In addition to the Security Program, with CPTMRO you have an easy to configure firewall PDS member.

Some of the command parameters documented in this guide reference data set members as the operands. This appendix describes the format of those members. All members must reside in the PDS (Partitioned Data Set) referred to by the PARMLIB DD startup JCL statement. The PARMLIB DD statement must reference a PDS.

SECURITY

The security member is used to specify remote IP address to limit access to the Listener object.

The keyword REMOTEIPADDR begins each line of the member followed by operands. Lines beginning with * denote comments. Continuation is denoted with a **- (**hyphen) as the last token on the line. The next line is then a continuation of the previous line.

REMOTEIPADDRESS	The REMOTEIPADDRESS statement is used to define a remote host (or group of hosts when a mask is used) that is recognized by the Listener. If the security member does not contain any IPADDRESS statements, then the Listener will allow all client access. Alias IPADDR `REMOTEIPADDRESS ipaddress [ MASK mask ] [ REJECT ] [ SESSION session ]`
ipaddress	Specifies a remote client IP address *nnn.nnn.nnn.nnn* where *nnn* is a decimal number in the range 0 to 255.
mask	Specifies a mask in the form of an IP address *nn.nnn.nnn.nnn, which* is "ANDed" with the incoming request and then compared to *ipaddress*. This lets clients from a semi-qualified host connect to this Listener. The default mask is 255.255.255.255, effectively limiting the foreign host to a single address. By specifying other masks, semi-qualified addresses can be set up for access to a given range of hosts.
REJECT	Optional. When specified, the given *ipaddress* or masked address is refused access to the Listener.
session	This is an override to the normal selection criteria of Session objects. This Listener will always use this session to forward the connect request.

Appendix B

Unicenter NetMaster Administrator Interface

This appendix introduces you to the Unicenter NetMaster Unicenter SOLVE:CPT Administrator Interface (Administrator Interface) panels. This Unicenter NetMaster product suite feature is available to all Unicenter SOLVE:CPT users that have the NetMaster product.

Starting the Interface

You can access the Administrator Interface via a menu shortcut. Type **/IPMON** on the command line and press Enter. Then type **?** before the CICS resource you want to view or for which you want to perform a task.

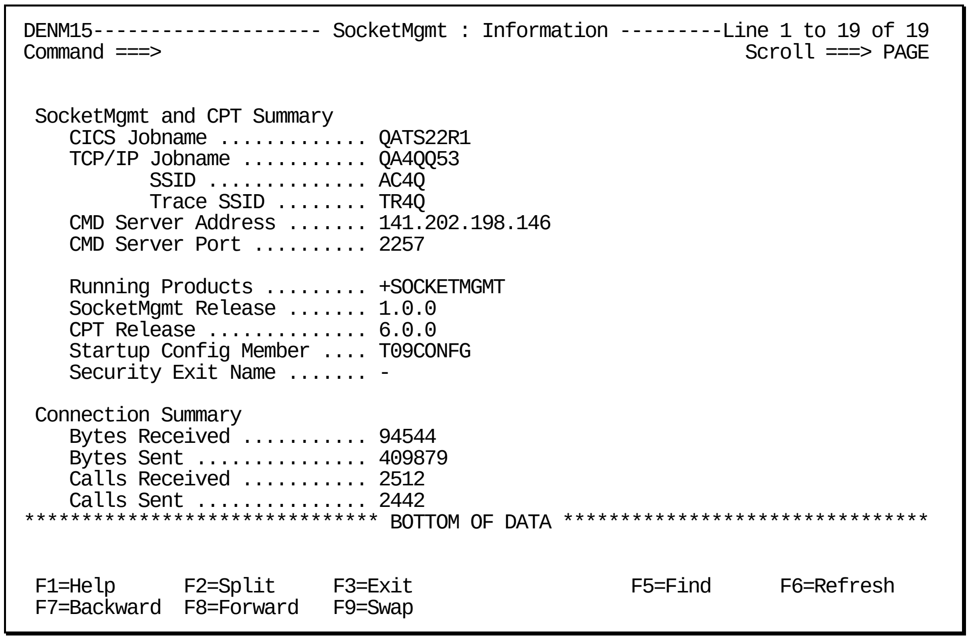
Moving Through the Panels

Use the commands on this panel to view and perform the task with which they are associated.

```
Command       Description
AL            View Alerts for a Resource
AM            Activate Monitoring
CL            Connections List via SocketMgmt
CMD           Command Entry - SocketMgmt
D             Display Resource Status
DUI           Delete SocketMgmt User Info. for Server
H             Show Performance History
IC            IP Connections
ICC           IP CICS Connections
IM            Inactivate Monitoring
SB            SocketMgmt and CPT Bounce
SQ            SocketMgmt Query Display
SS            CICS Server Start
SSB           SocketMgmt CMD Server Bounce
TS            CICS Transaction Start
UM            Update Resource Monitoring Definition
W             Display Outstanding WTORs for JOB/STC
```

Unicenter SOLVE:CPT Query Display (SQ)

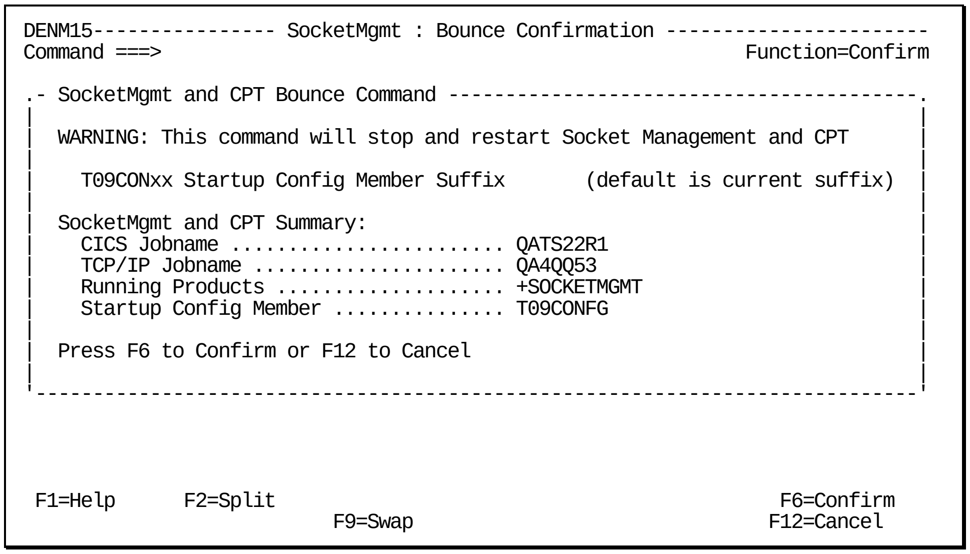
Use the SQ command to display a panel that enables you to view global information for the Unicenter SOLVE:CPT product in the CICS address space.



```
DENM15-------------------- SocketMgmt : Information ---------Line 1 to 19 of 19
Command ===>                                                    Scroll ===> PAGE


 SocketMgmt and CPT Summary
    CICS Jobname ............. QATS22R1
    TCP/IP Jobname ........... QA4QQ53
           SSID .............. AC4Q
           Trace SSID ........ TR4Q
    CMD Server Address ....... 141.202.198.146
    CMD Server Port .......... 2257

    Running Products ......... +SOCKETMGMT
    SocketMgmt Release ....... 1.0.0
    CPT Release .............. 6.0.0
    Startup Config Member .... T09CONFG
    Security Exit Name ....... -

 Connection Summary
    Bytes Received ........... 94544
    Bytes Sent ............... 409879
    Calls Received ........... 2512
    Calls Sent ............... 2442
******************************* BOTTOM OF DATA ********************************


 F1=Help      F2=Split     F3=Exit                   F5=Find      F6=Refresh
 F7=Backward  F8=Forward   F9=Swap
```

Socket Management and CPT Bounce (SB)

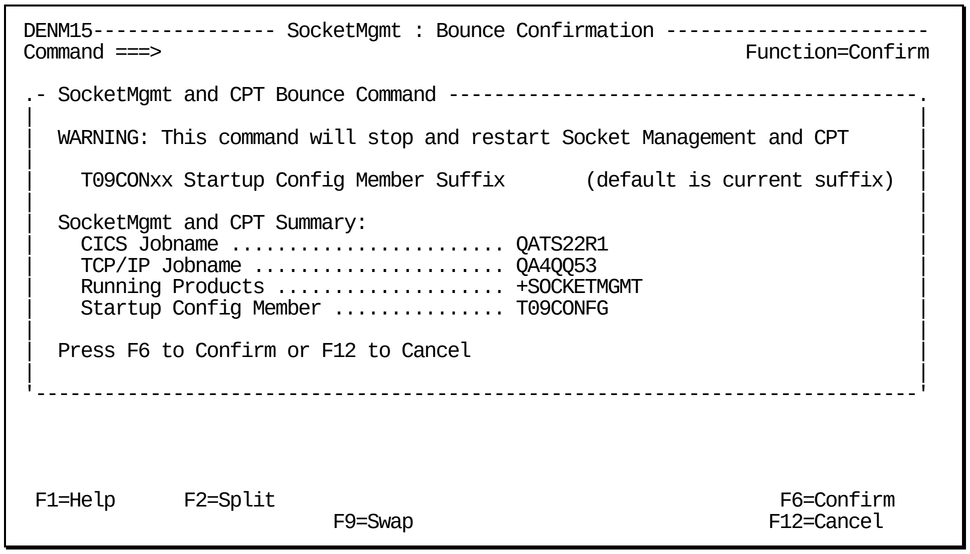
Use the SB command to shut down and restart Unicenter SOLVE:CPT in the CICS address space.



```
DENM15---------------- SocketMgmt : Bounce Confirmation -----------------------
Command ===>                                                    Function=Confirm

.- SocketMgmt and CPT Bounce Command ----------------------------------------.
|                                                                            |
|  WARNING: This command will stop and restart Socket Management and CPT     |
|                                                                            |
|    T09CONxx Startup Config Member Suffix       (default is current suffix) |
|                                                                            |
|  SocketMgmt and CPT Summary:                                               |
|    CICS Jobname ........................ QATS22R1                          |
|    TCP/IP Jobname ...................... QA4QQ53                           |
|    Running Products .................... +SOCKETMGMT                       |
|    Startup Config Member ............... T09CONFG                          |
|                                                                            |
|  Press F6 to Confirm or F12 to Cancel                                      |
|                                                                            |
'----------------------------------------------------------------------------'



 F1=Help      F2=Split                                            F6=Confirm
                           F9=Swap                               F12=Cancel
```

Unicenter SOLVE:CPT CMD Server Bounce (SSB)

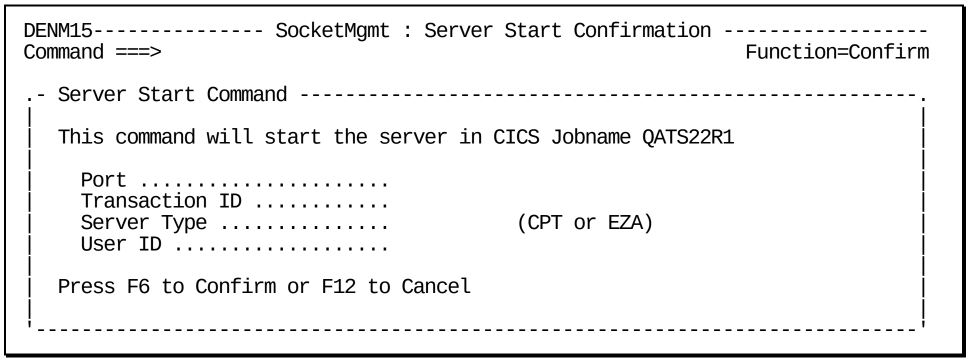
Use the SSB command to display a panel to stop and restart Unicenter SOLVE:CPT in the CICS address space.



```
DENM15---------------- SocketMgmt : Bounce Confirmation -----------------------
Command ===>                                                   Function=Confirm

.- SocketMgmt and CPT Bounce Command ----------------------------------------.
|                                                                            |
|  WARNING: This command will stop and restart Socket Management and CPT     |
|                                                                            |
|    T09CONxx Startup Config Member Suffix       (default is current suffix) |
|                                                                            |
|  SocketMgmt and CPT Summary:                                               |
|    CICS Jobname ........................ QATS22R1                           |
|    TCP/IP Jobname ...................... QA4QQ53                            |
|    Running Products .................... +SOCKETMGMT                        |
|    Startup Config Member ............... T09CONFG                           |
|                                                                            |
|  Press F6 to Confirm or F12 to Cancel                                      |
|                                                                            |
'----------------------------------------------------------------------------'



 F1=Help      F2=Split                                            F6=Confirm
                           F9=Swap                               F12=Cancel
```

CICS Server Start (SS)

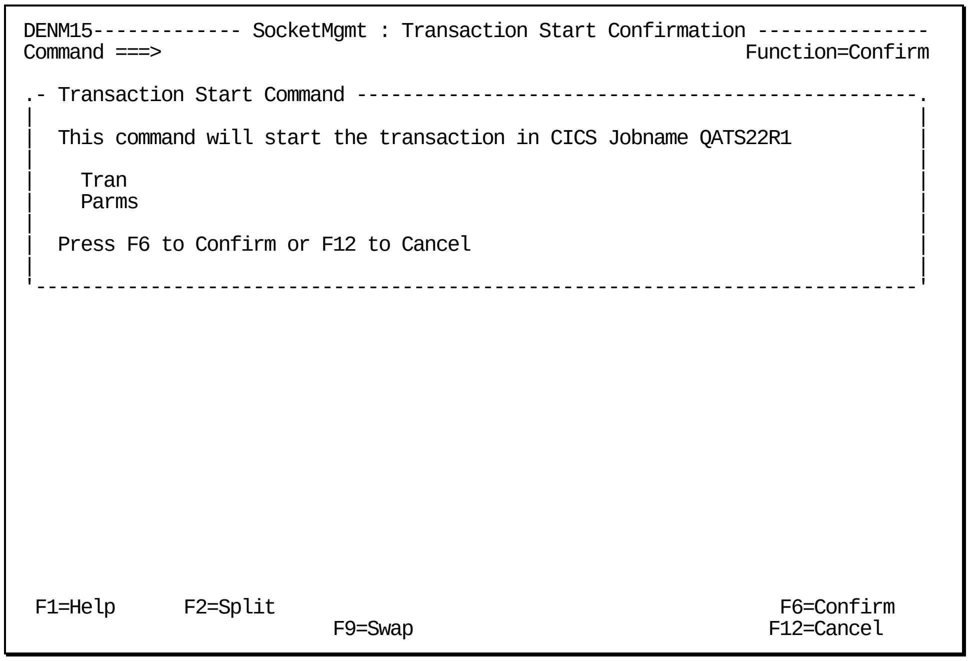
Use the SS command to start and EZASOKET server in the CICS address space.



```
DENM15--------------- SocketMgmt : Server Start Confirmation ------------------
Command ===>                                                   Function=Confirm

.- Server Start Command -----------------------------------------------------.
|                                                                            |
|  This command will start the server in CICS Jobname QATS22R1               |
|                                                                            |
|    Port ......................                                             |
|    Transaction ID ............                                             |
|    Server Type ...............          (CPT or EZA)                       |
|    User ID ...................                                             |
|                                                                            |
|  Press F6 to Confirm or F12 to Cancel                                      |
|                                                                            |
'----------------------------------------------------------------------------'
```

CICS Transaction Start (TS)

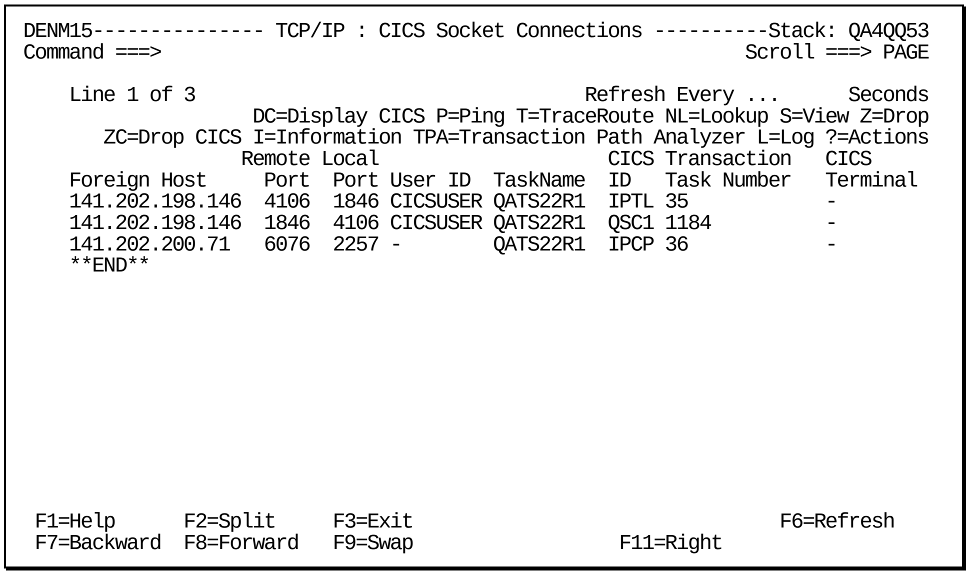
Use the TS command to start a transaction in the CICS address space.



```
DENM15------------ SocketMgmt : Transaction Start Confirmation ---------------
Command ===>                                                 Function=Confirm

.- Transaction Start Command -------------------------------------------------.
|                                                                             |
|  This command will start the transaction in CICS Jobname QATS22R1           |
|                                                                             |
|    Tran                                                                     |
|    Parms                                                                    |
|                                                                             |
|  Press F6 to Confirm or F12 to Cancel                                       |
|                                                                             |
'-----------------------------------------------------------------------------'










 F1=Help      F2=Split                                          F6=Confirm
                          F9=Swap                              F12=Cancel
```

IP CICS Connections (ICC)

Use the ICC command to display session information.

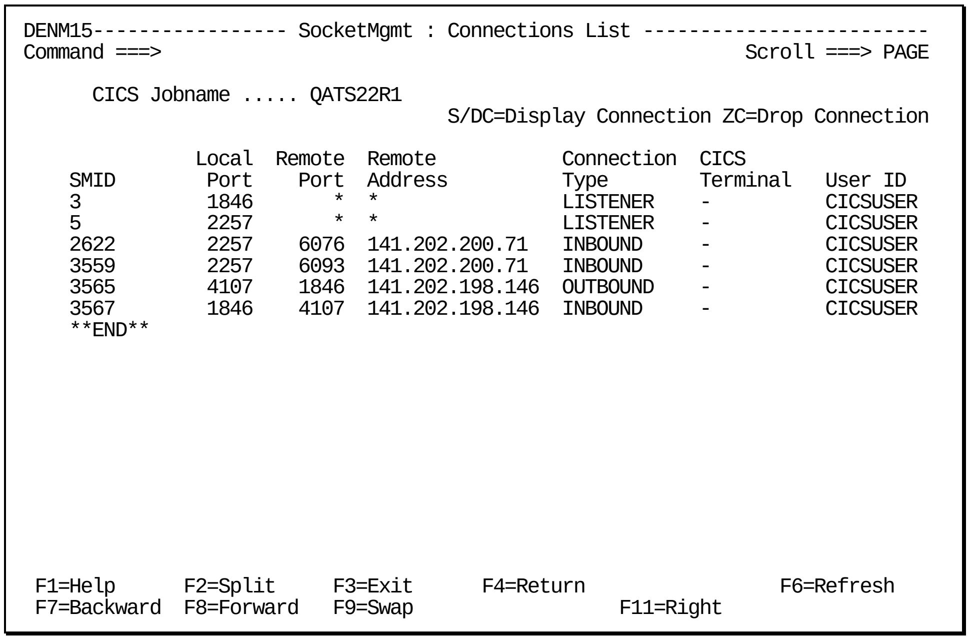


```
DENM15--------------- TCP/IP : CICS Socket Connections ----------Stack: QA4QQ53
Command ===>                                                 Scroll ===> PAGE

    Line 1 of 3                                      Refresh Every ...      Seconds
                    DC=Display CICS P=Ping T=TraceRoute NL=Lookup S=View Z=Drop
       ZC=Drop CICS I=Information TPA=Transaction Path Analyzer L=Log ?=Actions
                   Remote Local                        CICS Transaction   CICS
    Foreign Host     Port   Port User ID  TaskName  ID   Task Number   Terminal
    141.202.198.146  4106  1846 CICSUSER QATS22R1  IPTL 35             -
    141.202.198.146  1846  4106 CICSUSER QATS22R1  QSC1 1184           -
    141.202.200.71   6076  2257 -        QATS22R1  IPCP 36             -
    **END**









 F1=Help      F2=Split     F3=Exit                              F6=Refresh
 F7=Backward  F8=Forward   F9=Swap                  F11=Right
```

Connections List (CL)

Page 1

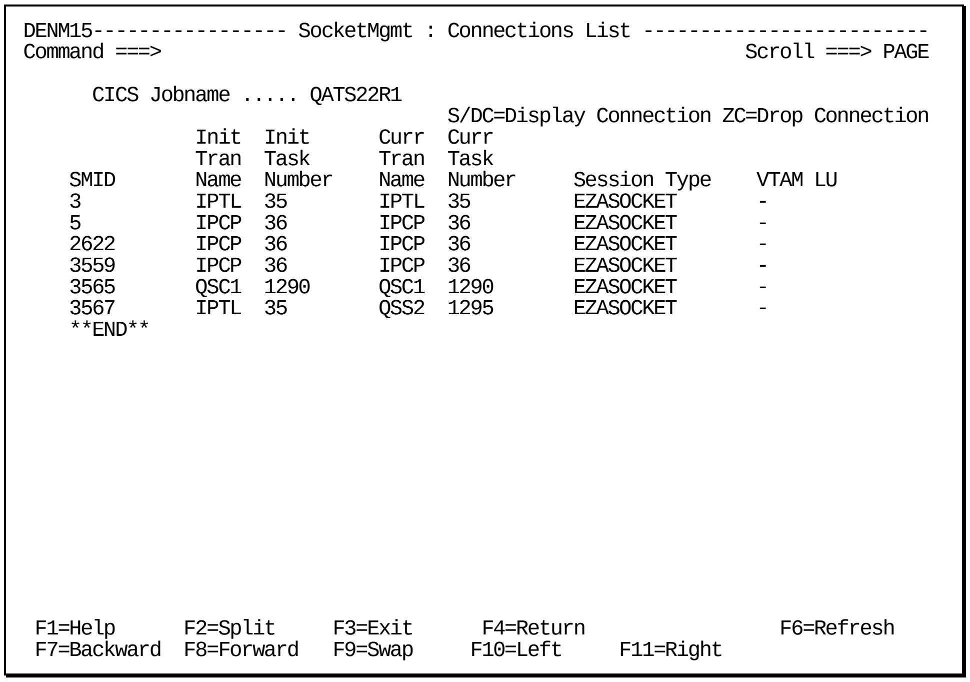
Use the CL command to display active connection information in the CICS address space.



```
DENM15----------------- SocketMgmt : Connections List -------------------------
Command ===>                                                      Scroll ===> PAGE

      CICS Jobname ..... QATS22R1
                                     S/DC=Display Connection ZC=Drop Connection

               Local  Remote  Remote           Connection  CICS
    SMID        Port    Port  Address          Type        Terminal   User ID
    3           1846       *  *                LISTENER    -          CICSUSER
    5           2257       *  *                LISTENER    -          CICSUSER
    2622        2257    6076  141.202.200.71   INBOUND     -          CICSUSER
    3559        2257    6093  141.202.200.71   INBOUND     -          CICSUSER
    3565        4107    1846  141.202.198.146  OUTBOUND    -          CICSUSER
    3567        1846    4107  141.202.198.146  INBOUND     -          CICSUSER
    **END**









 F1=Help      F2=Split     F3=Exit      F4=Return                 F6=Refresh
 F7=Backward  F8=Forward   F9=Swap                  F11=Right
```

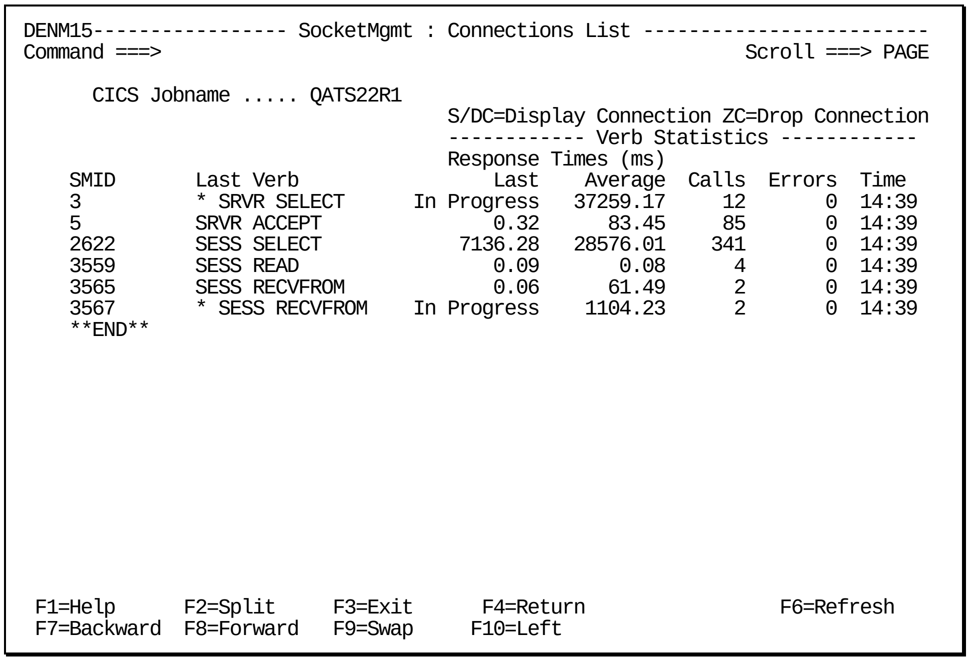
Note: You can select a specific session to view more information.

Command List (CL)
Page 2



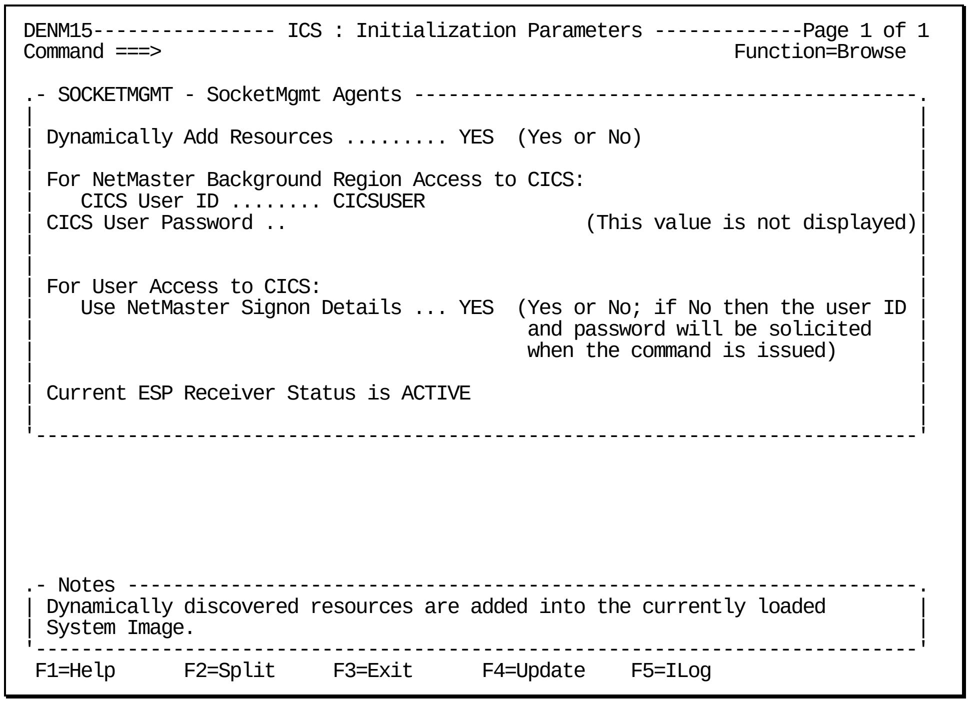
```
DENM15----------------- SocketMgmt : Connections List -------------------------
Command ===>                                                  Scroll ===> PAGE

     CICS Jobname ..... QATS22R1
                                     S/DC=Display Connection ZC=Drop Connection
              Init  Init      Curr  Curr
              Tran  Task      Tran  Task
   SMID       Name  Number    Name  Number     Session Type    VTAM LU
   3          IPTL  35        IPTL  35         EZASOCKET       -
   5          IPCP  36        IPCP  36         EZASOCKET       -
   2622       IPCP  36        IPCP  36         EZASOCKET       -
   3559       IPCP  36        IPCP  36         EZASOCKET       -
   3565       QSC1  1290      QSC1  1290       EZASOCKET       -
   3567       IPTL  35        QSS2  1295       EZASOCKET       -
   **END**

 F1=Help     F2=Split     F3=Exit      F4=Return                 F6=Refresh
 F7=Backward F8=Forward   F9=Swap     F10=Left     F11=Right
```

Command List (CL)
Page 3



```
DENM15----------------- SocketMgmt : Connections List -------------------------
Command ===>                                                  Scroll ===> PAGE

     CICS Jobname ..... QATS22R1
                                     S/DC=Display Connection ZC=Drop Connection
                                     ------------ Verb Statistics ------------
                                     Response Times (ms)
   SMID       Last Verb                  Last     Average  Calls  Errors  Time
   3          * SRVR SELECT      In Progress    37259.17     12       0  14:39
   5          SRVR ACCEPT               0.32       83.45     85       0  14:39
   2622       SESS SELECT            7136.28    28576.01    341       0  14:39
   3559       SESS READ                 0.09        0.08      4       0  14:39
   3565       SESS RECVFROM             0.06       61.49      2       0  14:39
   3567       * SESS RECVFROM    In Progress     1104.23      2       0  14:39
   **END**

 F1=Help     F2=Split     F3=Exit      F4=Return                 F6=Refresh
 F7=Backward F8=Forward   F9=Swap     F10=Left
```

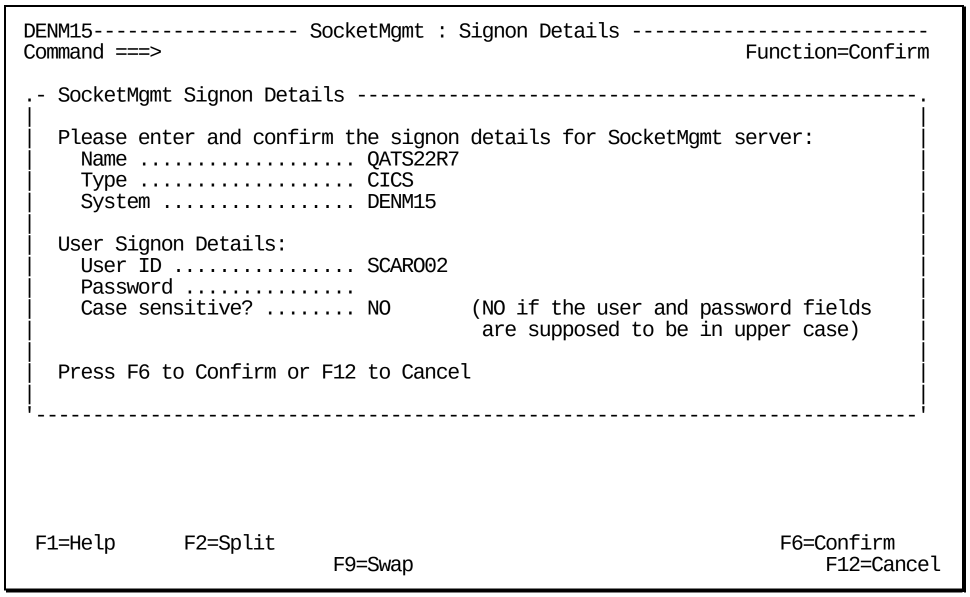
Initialization and Customization Services Parameter (admin function)



```
DENM15---------------- ICS : Initialization Parameters -------------Page 1 of 1
Command ===>                                                     Function=Browse

.- SOCKETMGMT - SocketMgmt Agents ------------------------------------------.
|                                                                            |
| Dynamically Add Resources ......... YES  (Yes or No)                       |
|                                                                            |
| For NetMaster Background Region Access to CICS:                            |
|    CICS User ID ........ CICSUSER                                          |
| CICS User Password ..                            (This value is not displayed)|
|                                                                            |
|                                                                            |
| For User Access to CICS:                                                   |
|    Use NetMaster Signon Details ... YES  (Yes or No; if No then the user ID |
|                                           and password will be solicited   |
|                                           when the command is issued)      |
|                                                                            |
| Current ESP Receiver Status is ACTIVE                                      |
|                                                                            |
'----------------------------------------------------------------------------'





.- Notes --------------------------------------------------------------------.
| Dynamically discovered resources are added into the currently loaded       |
| System Image.                                                              |
'----------------------------------------------------------------------------'
 F1=Help      F2=Split     F3=Exit      F4=Update    F5=ILog
```

Signon Details

Use this panel to sign on to the command server. This enables you to display information from the CICS address space where Unicenter SOLVE:CPT is running.



```
DENM15------------------ SocketMgmt : Signon Details --------------------------
Command ===>                                                  Function=Confirm

.- SocketMgmt Signon Details ------------------------------------------------.
|                                                                            |
|  Please enter and confirm the signon details for SocketMgmt server:        |
|    Name ................... QATS22R7                                       |
|    Type ................... CICS                                           |
|    System ................. DENM15                                         |
|                                                                            |
|  User Signon Details:                                                      |
|    User ID ................ SCARO02                                        |
|    Password ...............                                                |
|    Case sensitive? ........ NO       (NO if the user and password fields   |
|                                        are supposed to be in upper case)   |
|                                                                            |
|  Press F6 to Confirm or F12 to Cancel                                      |
|                                                                            |
'----------------------------------------------------------------------------'




 F1=Help      F2=Split                                          F6=Confirm
                          F9=Swap                                   F12=Cancel
```

Appendix

C

UNIX Test Programs

This appendix contains installation, usage instructions, and the UNIX-style man pages for sample client and server programs that run on a UNIX platform. These programs can be used to remotely verify the installation of Unicenter SOLVE:CPT. See Test Programs for a brief description of how to test from either side of the connection.

The following topics are discussed in this appendix:

- Test Programs—Describes the test programs available.
- Installation Steps Summary—Summarizes the order to follow the sections needed to install and initially test the UNIX test programs
- Installing CICS TD UNIX Programs Using FTP—Describes how to move the code samples from the CPT distribution library on MVS to the UNIX system you will be installing them on.
- Building the Programs—Describes how the scripts setup to do the makefile to make the test programs executable.
- Testing the TCP Programs— Describes how to run the test programs using the TCP protocol
- Testing the UDP Programs— Describes how to run the test programs using the UDP protocol
- Installing the man Pages—Describes the procedure to do a standard UNIX install of the programs with man pages and uninstall setup.
- CL1 (1)—cl1: CICS transient data server test client
- SV1(1)—sv1: CICS transient data server test server
- UDPCLT (1)—UDPCLT: UDP test client program
- UDPSVR (1)—UDPSVR: UDP test server program

Note: For a full description and installation considerations for Unicenter SOLVE:CPT distributed samples for CICS, see the *Programmers Guide*.

Test Programs

There are two sets of programs:

- The cl1 and sv1 program set uses TCP as their transport

  sv1 —provides the TCP server function and listens for incoming connection requests.

  cl1 —is the TCP client and initiates a connection to the server.

- The udpclt and udpsvr program set uses UDP as their transport

  udpsvr —provides the UDP server function and waits for UDP datagrams.

  udpclt is the UDP client and sends a datagram to the server.

Review the following UNIX style man pages to familiarize yourself with the options and parameters accompanying the program calls.

There are several ways to run the programs with Unicenter SOLVE:CPT. The default runs against loopback. Alternatively, you can use the MVS sample programs, CL1, SV2, and SV3 to actively set up TCP connections. These samples are contained in the T09SAMP distribution library and are described in the P*rogrammers Guide*. For example, to run the TCP server on UNIX and the TCP client on MVS, modify the CL1 program on MVS to point to your UNIX host and port number.

Note: The default port number for the UNIX server is 2002. Next, compile and link CL1 for execution in CICS. Then, run the server on UNIX by typing:

```
sv1 -p xxx
```

where *xxx* is the port number you selected in the CL1 program. Run the CL1 program on MVS and verify the output is received correctly at the server.

You can also run the server on MVS. Default ports for the listener program are set differently, depending on the program language:

Language	Default Port
ASM	1783
SAS/C	1683
COBOL	1983
PL/1	1583

Compile, link, and execute SV2 and SV3 on MVS. Execute the client on UNIX with the following command:

```
cl1 -p xxx hostname
```

where *xxx* is the port number that the MVS program SV2 is listening on. Verify that characters typed at the UNIX client are received at the MVS server.

Installation Steps Summary

This section summarizes the sequence of sections that should be followed to properly install the UNIX test programs for Unicenter SOLVE:CPT.

1. Read the previous section, Test Programs, to get an understanding of the purpose of these test programs.
2. Follow the instructions in Installing CICS TD UNIX Programs Using FTP later in this chapter.
3. Follow the instructions in Building the Programs later in this chapter.
4. Follow the instructions in Testing the TCP Programs later in this chapter.
5. Follow the instructions in Testing the UDP Programs later in this chapter.
6. Follow the instructions in Installing the man Pages later in this chapter.
7. Follow the guidelines Test Programs earlier in this chapter to test Unicenter SOLVE:CPT connections bi-directionally.

Installing CICS TD UNIX Programs Using FTP

The CICS TD UNIX test programs are contained in an MVS PDS file T09SAMP unloaded as part of a standard install from the Unicenter SOLVE:CPT distribution tape. The T09SAMP PDS contains a shell script and the other program sources, make file, man pages, and so forth. The shell script is first downloaded to UNIX and then run to retrieve the rest of the data.

The examples in this section all user input is bold and all UNIX responses, prompts, and messages are indented in regular type following user input.

1. Create a UNIX directory to be used to copy and build the CICS TD UNIX programs (for these examples, the directory name is cics):

   ```
   mkdir cics
   ```

2. Change into the newly created directory:

   ```
   cd cics
   ```

3. Verify the working directory:

   ```
   pwd
   ```

   ```
   .../cics
   ```

4. Copy the shell script file to UNIX using FTP from the UNIX machine (for these examples, the download shell script name is ftp_from_mvs.sh):

   ```
   ftp mvs_host_name
     Connected to mvs_host_name.
     220 mvs_host_name -- FTP Server, Enter command or HELP
     Name (mvs_host_name:unix_userid): mvs_userid
     331 Enter PASS command
     Password:mvs_password
     230 Logged in -  Host mvs_host_name  User mvs_userid  Sess# num
     ftp> get 'cpt.high.level.quilifier.T09SAMP(cptusftp)' ftp_from_mvs.sh
     200 OK, Ready
     150-Dataset open with attributes:
     Type A N    Tabs    8   Stru F   Mode S
     Path cpt.high.level.quilifier.T09SAMP(CPTUSFTP)  Volser volume  Unit unit
     Dsorg   PO   Recfm FB   Lrecl   80   Blksize 6160
     150
     226-Transfer complete
     Sess# num   7959 bytes sent in 2.07 seconds (3844 bytes/s)
     Path cpt.high.level.quilifier.T09SAMP(CPTUSFTP)  User mvs_userid  Data
   bytes sent 35440
     Disk tracks read 2
     226
     local: ftp_from_mvs_host_name remote:
   cpt.high.level.quilifier.T09SAMP(cptusftp)
     7959 bytes received in 1.7 seconds (4.6 Kbytes/s)
     ftp> quit
     221 Session terminated
   ```

5. Make the shell script executable:

   ```
   chmod +x ftp_from_mvs.sh
   ```

6. Verify that the execute permission bit(s) have been set for the file (use the -l option):

   `ls -l ftp_from_mvs.sh`

   ```
   -rwxrwx---  1 uid 7748 Oct  9 17:10 ftp_from_mvs.sh
   ```

7. Execute the shell script and respond to the prompts. The script FTPs the rest of the CICS TD UNIX program sources, make file, man pages, and so forth to UNIX.

   Enter these commands:

   ```
   ftp_from_mvs.sh
    Enter MVS data set name for CICS UNIX programs data set
    (fully-qualified without quotes): cpt.high.level.quilifier.T09SAMP
    Enter MVS host name for FTP: mvs_host_name
    Enter MVS userid for FTP: mvs_userid
    Enter password for MVS userid 'mvs_userid': mvs_password
    Do you want an FTP log created? (y/n): n
    MVS file name (quotes added by script): 'cpt.high.level.quilifier.T09SAMP'
    MVS host name for FTP: mvs_host_name
    MVS userid for FTP: mvs_userid
    FTP log to be created: n
    The following target files will be deleted and recreated by this
    process:
    Makefile README cl1.1 cl1.c
    sv1.1 sv1.c util.c xlate.c
    udpclt.1 udpclt.c udpsvr.1 udpsvr.c
    man1
    Are these values OK? (y/n):y
    All target files were FTP'ed or created successfully
   ```

   The script indicates that the files were FTPed successfully.

   Note:

   - If the script fails, run it with the -d (debug) option and request that an FTP log be made. Review the debug messages and the FTP log file to determine what the problem is.
   - The script can be rerun as many times as necessary without having to redo previous steps in this procedure.
   - If you prefer to do the FTPs manually rather than using the script, do the following additional FTP commands as part of Step 4:

     ```
     get 'cpt.high.level.quilifier.T09SAMP(t09ukmak)' Makefile
     get 'cpt.high.level.quilifier.T09SAMP(t09uread)' README
     get 'cpt.high.level.quilifier.T09SAMP(t09umcl1)' cl1.1
     get 'cpt.high.level.quilifier.T09SAMP(t09uccl1)' cl1.c
     get 'cpt.high.level.quilifier.T09SAMP(t09umsv1)' sv1.1
     get 'cpt.high.level.quilifier.T09SAMP(t09ucsv1)' sv1.c
     get 'cpt.high.level.quilifier.T09SAMP(t09ucutl)' util.c
     get 'cpt.high.level.quilifier.T09SAMP(t09ucxlt)' xlate.c
     get 'cpt.high.level.quilifier.T09SAMP(t09umucl)' udpclt.1
     get 'cpt.high.level.quilifier.T09SAMP(t09ucucl)' udpclt.c
     get 'cpt.high.level.quilifier.T09SAMP(t09umusv)' udpsvr.1
     get 'cpt.high.level.quilifier.T09SAMP(t09ucusv)' udpsvr.c
     ```

Building the Programs

Run the UNIX make program to build the CICS TD UNIX programs from the sources in the directory (the all option builds both the client and the server CICS TD programs).

If you are making these commands on a Solaris workstation, you have to edit the Makefile to include the proper libraries. The Makefile includes the instructions to do this.

```
make all

 cc -Dunix   -target sun4 -c cl1.c
 cc -Dunix   -target sun4 -c util.c
 cc -Dunix   -target sun4 -c xlate.c
 cc -Dunix    cl1.o util.o xlate.o -o cl1
 cl1 (TCP client) binary has been built.
 cc -Dunix   -target sun4 -c sv1.c
 cc -Dunix    sv1.o util.o xlate.o -o sv1
 sv1 (server) binary has been built.
 cc -Dunix   -target sun4 -c udpclt.c
 cc -Dunix    udpclt.o -o udpclt
 udpclt (UDP client) binary has been built.
 cc -Dunix   -target sun4 -c udpsvr.c
 cc -Dunix    udpsvr.o -o udpsvr
 udpsvr (UDP server) binary has been built.
```

The make output should indicate that the client and server program binaries were successfully built.

Testing the TCP Programs

Use this series of commands to verify operation of the TCP programs in a loopback mode:

```
sv1 -o all -f x &
cl1 -o all -f cl1
cmp cl1 x
```

This runs the server (sv1) in the background and directs the server to receive data into file x using the transfer option all (receive all data as is). The server is started before the client so that it is running when the client connects to it.
The client is started using the same transfer mode (-o all) and directed to transfer file cl1 (the client binary file) to file x. Then the UNIX cmp program compares the input and output files. If cmp exits silently, the two files are the same.

Testing the UDP Programs

1. Start the server with the following command:

   ```
   udpsvr >& logname &
   ```

   This runs the server in background and writes to the file *logname*.

2. Start the client with the following command, and when prompted, type in a message and press <Return>.

   ```
   udpclt
   ```

   ```
   UDP Client> Address data is now stored in ‘local’ & ‘remote’
   UDP Client> Enter message
   Hello UDP world.
   ```

   The following will be returned to the client:

   ```
   UDP Client> sent message

   UDP Client> The returned message is:  Hello UDP world.
   UDP Client> Its length is          :  16
   UDP Client> socket closed.
   ```

   The server exits after the message is returned.

3. To look at the server’s output, type:

   ```
   cat logname
   ```

   ```
   UDP Server> Address is stored in ‘local’
   UDP Server> Waiting at port # 12345
   UDP Server> length of the message is: 16
   UDP Server> the message is: Hello UDP world.
   UDP Server> sent the message back
   UDP Server> Socket closed.
   ```

Installing the man Pages

The make file can also install the binaries and man pages into system directories. This step is optional as the programs can be run from the directory in which they were created. Before running the make install procedure, review the make file to see if the target directories for the binaries and man pages are suitable for your installation. They should be /usr/local for the binaries and /usr/man/man1 for the man pages. If these are not acceptable, then you can either edit the make file (Makefile) and change the directory names in the BINDIR and MANDIR macro definitions, or override the values when make is run.

Note: There are different install commands depending on the platform that you are using.

```
make install         for SunOS based systems including DEC/Ultrix

make installhp       for HP based systems
make installsvr4     for SVR4 systems including RS/6000 and Solaris
```

There is also a special call, uninstallhp, to remove the executable code from HP systems.

To run make with the default directories, or if you have changed them in the make file, enter these commands:

Note: These commands usually require root privileges.

```
su

 Password: root_password

make install
 install -s cl1     /usr/local
 install -s sv1     /usr/local
 install -s udpclt     /usr/local
 install -s udpsvr     /usr/local
 install cl1.1     /usr/man/man1
 install sv1.1     /usr/man/man1
 install udpclt.1     /usr/man/man1
 install udpsvr.1     /usr/man/man1
 cl1/sv1 and udpclt/udpsvr binaries and man pages have been
  installed.
exit
```

To override the directory specifications, enter this command:

```
make install BINDIR=binaries_dir MANDIR=man_pages_dir
```

In either case, the make execution displays messages showing the commands issued and indicating that things went successfully.

After the make install process, you can run make clean to cleanup the object and binaries from the CICS TD UNIX programs directory:

```
make clean
 rm -f *.o cl1 sv1 udpclt udpsvr
 cl1/sv1 udpclt/udpsvr binaries and object files have been
  deleted.
```

The effect of an install can be undone with make uninstall:

```
su
 Password:root_password
make uninstall
 rm -f /usr/local/cl1         /usr/local/sv1
 rm -f /usr/man/man1/cl1.1         /usr/man/man1/sv1.1
 rm -f /usr/man/cat1/cl1.1         /usr/man/cat1/sv1.1
 rm -f /usr/local/udpclt         /usr/local/udpsvr
 rm -f /usr/man/man1/udpclt.1         /usr/man/man1/udpsvr.1
 rm -f /usr/man/cat1/udpclt.1         /usr/man/cat1/udpsvr.1
 cl1/sv1 and udpclt/udpsvr binaries and man pages have been
  de-installed.
exit
```

CL1 (1)

cl1—CICS Transient Data Server Test Client

```
cl1[ -dt ] [ -o option ] [ -s sep ] [ -f filename ]
[ -p port ] [ -x traceval ] [ host ]
```

where:

cl1	Transmits the contents of the standard input or filename to the CICS transient data server at the specified host and port using the specified option. Operates silently unless an error is encountered or the -d (debug) and/or -x (hex trace) switches are specified. The program returns 0 if successful; otherwise, it returns 1. The source for cl1 is cl1.c. The program also calls common functions in util.c and xlate.c.
-d	Turn on debugging mode for more verbose output. In this mode, arguments are echoed, socket function completions are noted, IP and port addresses are echoed when the connection to the server is made, and statistics are displayed at the end of the transfer. The default is not to use debugging mode.
-t	Translate the file data from ASCII to EBCDIC. The translate table used is table *atoe* in xlate.c. The default is not to translate.
-x *traceval*	Generate a hexadecimal trace of input or output data. A traceval of 0 means not to trace, 1 means to trace input data received from the network, 2 means to trace output data written to filename or to standard output, and 3 means to trace both input and output data. The default is not to trace.
-o *option*	Use the specified transfer mode option. The following options are supported: SEP, LL, FILE, and ALL. The default option is SEP. The case is which option is specified does not matter.

sep	Indicates that each record of the input file is to be transferred with a separator sequence appended to the end of the record; the sequence is specified as sep. A record is defined as that portion of the input data up to, but not including, the terminating ASCII LF character.
LL	Indicates that each record is to be preceded by a two-byte length field; the length specified in the length field is the actual record length and does not include the two-byte length field itself. A record is defined as above.
FILE	Indicates that the data is to be sent *as is* but limited to 32767 or less bytes.
ALL	Indicates that all of the data is to be sent *as is*.

-s *sep*	When transferring with the SEP option, use the specified sep sequence to terminate each record. The following separator sequences are supported: CRLF, CR, LF and hexval. The default separator sequence is CRLF. This switch is only valid when the SEP option is specified or defaulted. The case in which sep is specified does not matter.	
	CRLF	Indicates that each record is to be terminated with an ASCII CRLF sequence (0x0D0A).
	CR	Indicates that each record is to be terminated with an ASCII CR character (0x0D).
	LF	Indicates that each record is to be terminated with an ASCII LF character (0x0A).
	hexval	Indicates that the specified hexadecimal string is to appended to the end of each record. hexval must specify an even number of hex digits and must begin with the sequence 0x or 0X.
-f *filename*	Read the input from the specified filename instead of from the standard input.	
-p *port*	Connect to the specified port. The default port is nine (discard).	
host	Connect to the CICS server at host. The default host is “127.0.0.1” (loopback).	

SV1(1)

sv1—CICS Transient Data Server Test Server

```
sv1 [ -dt ] [ -o option ] [ -s sep ] [ -f filename ]
[ -p port ] [ -x traceval ]
```

where:

sv1	listens on port and, when connected to, receives data from a CICS transient data client into filename or to the standard output using the specified option. Operates silently unless an error is encountered or the -d (debug) and/or -x (hex trace) switches are specified. The program returns 0 if successful; otherwise, it returns 1. The source for sv1 is sv1.c. The program also calls common functions in util.c and xlate.c.
-d	Turn on debugging mode for more verbose output. In this mode, arguments are echoed, socket function completions are noted, IP and port addresses are echoed when the connection to the server is made, and statistics are displayed at the end of the transfer. The default is not to use debugging mode.
-t	Translate the network data from EBCDIC to ASCII. The translate table used is table “etoa” in xlate.c. The default is not to translate.
-x *traceval*	Generate a hexadecimal trace of received input data or written output data. A traceval of 0 means to generate no trace, 1 means to trace input data received from the network, 2 means to trace output data written to filename or standard output, and 3 means to trace both input and output data. The default is not to trace any input or output data.
-o *option*	Use the specified transfer mode option. The following options are supported: SEP, LL, FILE, and ALL. The default option is SEP. The case is which option is specified does not matter.

SEP	Indicates that each network will be terminated by a separator sequence that will appended to the end of the record; the sequence is specified as sep. Each record will have the separator sequence deleted and a LINEFEED appended before being written to the local file.
LL	Indicates that a two-byte length field will precede each network record; the length specified in the length field is the actual record length and does not include the two-byte length field itself. Each record will have the LL field stripped and a LINEFEED appended before being written to the local file.

FILE	Indicates that the data is to be received "as is" but limited to 32767 or less bytes (since supposedly the record is one CICS transient data record and such records are limited to a maximum of 32767 bytes). The network data will be written to the local file without modification.
ALL	Indicates that all of the network data is to be received *as is.* The network data will be written to the local file without modification

-s *sep*

When receiving network data with the SEP option, use the specified sep sequence to terminate each record. The following separator sequences are supported: CRLF, CR, LF, and hexval. The default separator sequence is CRLF. This switch is only valid when SEP option is specified or defaulted. The case in which sep is specified does not matter.

CRLF	Indicates that each record will be terminated with an ASCII CRLF sequence (0x0D0A).
CR	Indicates that each record will be terminated with an ASCII CR character (0x0D).
LF	Indicates that each record will be terminated with an ASCII LF character (0x0A).
hexval	Indicates that the specified hexadecimal string is to appended to the end of each record. hexval must specify an even number of hex digits and must begin with the sequence 0x or 0X.

-f *filename*

Write the received data to the specified filename instead of to the standard output.

-p *port*

Listen on the specified port.

Default port: 2002.

UDPCLT (1)

udpclt—UDP Test Client Program

```
udpclt [ hostname ] [ port ]
```

where:

udpclt	Prompts the user for a message, then sends the datagram via UDP to the specified *port* on *hostname* and receives the message back from the transient data server. The server also returns the length of the returned message. (See also Test Programs.) The source for udpclt is udpclt.c.
-hostname	Destination host. The default is the local hostname.
-port	The port number on which the UDP transient server is awaiting datagrams. The default port number is 12345.

UDPSVR (1)

udpsvr—UDP test server program

```
udpsvr [ port ]
```

where:

udpsvr	Waits on the specified *port* and, when a datagram is received, echoes it back to the client. The length of the message is also returned. (See also Test Programs.) The source for udpsvr is udpsvr.c.
-port	The port number on which the server awaits datagrams. The port number must be a positive integer. The default port number is 12345.

Appendix D

The Terminal Administrator Interface

This appendix describes the CICS/TS transactions and panels of the Unicenter SOLVE:CPT Administrator Interface (the Interface).

It discusses the following topics:

- Utilization—Describes monitoring, tracing, and termination of TCP connections
- Configuration Table Parameters panel—Describes how to display, update, and add Unicenter SOLVE:CPT Tool definitions
- Operations—Describes startup and graceful termination of the Unicenter SOLVE:CPT Interface, and control of the diagnostic tool and statistics

The Unicenter SOLVE:CPT Administrator Interface is a set of CICS/TS transactions and screens that improve visibility and control of the TCP/IP connections being established by the Interface. It provides a system administrator with the tools to trace, diagnose, and terminate any CICS/TS transaction using TCP or UDP calls.

The Administrator Interface also enables dynamic modification of the Unicenter SOLVE:CPT tools defined in the Configuration Table, and the ability to create new Unicenter SOLVE:CPT tool definitions.

Administrator Interface feature are grouped into the following major components:

Utilization:	Monitoring, tracing, and termination of TCP connections.
Configuration:	Display, update, and addition of Configuration Table entries.
Operations:	Start up and graceful termination of the Interface, and control of diagnostic tools and statistics.

The following sections describe each of the CICS/TS screens (called panels) used by the Administrator Interface, and the services they provide to a system administrator.

Entering the Administrator Interface

The Main Menu for the Administrator Interface is invoked with the IPAM transaction. This transaction produces the Main Menu display as shown in this example:

Note: All the Administrator Interface transactions have a three-character prefix of IPA. You can change this default at installation if these transaction IDs conflict with existing CICS/TS *transid*s. (The default IPAM transaction is given here.)

The Main Menu contains selections for the three main entry points into the Administrator Interface: Utilization, Configuration, and Operations. There is also a selection for Navigation that describes the standard PF key and function code assignments.

```
                    WELCOME TO THE

                 COMPUTER ASSOCIATES

       Unicenter NetMaster/Unicenter SOLVE:CPT
               ADMINISTRATOR INTERFACE


                PF4)  U: UTILIZATION

                PF5)  C: CONFIGURATION

                PF6)  O: OPERATIONS

                PF9)  K: NAVIGATION
```

Function Codes and PF Keys

You can make a selection on the Main Menu:

- Using a PF key
- Entering a function code (a one- or two-character command typed on the command line at the bottom of most Administrator Interface panels)

For example, to go to the first Utilization panel from the Main Menu, you can either press the PF4, or type **U** on the command line and press Enter.

The command line in a panel is identified by the **>** symbol. The command line is a 12-byte input field, followed by an error message field on the same line.

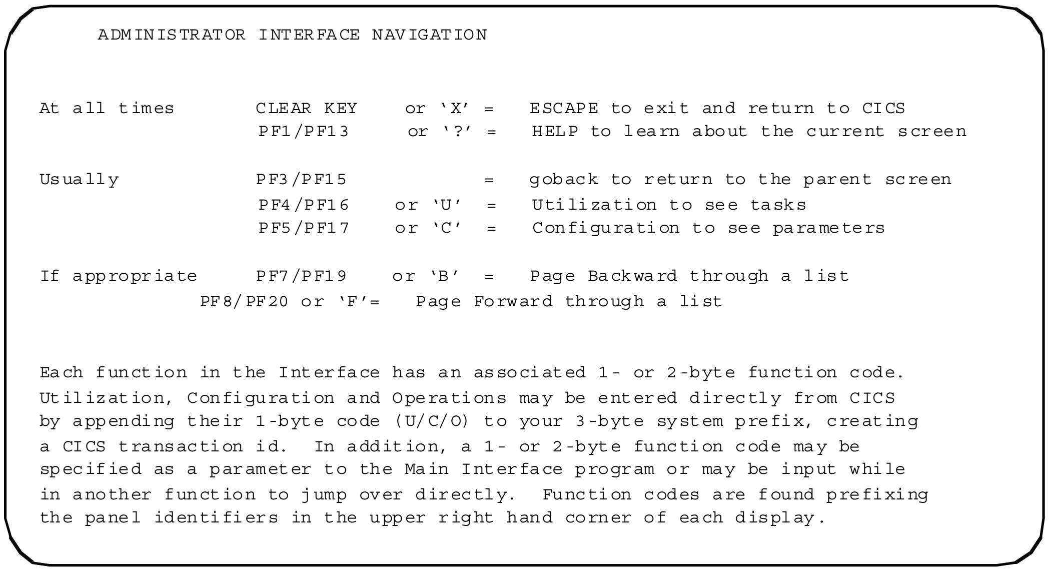
Every panel in the Administrator Interface makes this combination of PF keys and function codes available to you for navigating from panel to panel. The PF key choices appear at the bottom of each panel to assist novice users, while the function codes are available for experienced users who want to jump directly to a specific panel.

Note: Function assignments for PF keys 13–24 are always equivalent to the assignment of PF keys 1–12.

Administrator Interface Navigation

The fourth selection, Navigation, on the Main Menu brings up the online explanation of the standard PF key values and navigation function codes.

The following is a sample panel.



```
        ADMINISTRATOR INTERFACE NAVIGATION


  At all times      CLEAR KEY   or 'X' =  ESCAPE to exit and return to CICS
                    PF1/PF13    or '?' =  HELP to learn about the current screen

  Usually           PF3/PF15           =  goback to return to the parent screen
                    PF4/PF16   or 'U'  =  Utilization to see tasks
                    PF5/PF17   or 'C'  =  Configuration to see parameters

  If appropriate    PF7/PF19   or 'B'  =  Page Backward through a list
               PF8/PF20 or 'F'=   Page Forward through a list


  Each function in the Interface has an associated 1- or 2-byte function code.
  Utilization, Configuration and Operations may be entered directly from CICS
  by appending their 1-byte code (U/C/O) to your 3-byte system prefix, creating
  a CICS transaction id.  In addition, a 1- or 2-byte function code may be
  specified as a parameter to the Main Interface program or may be input while
  in another function to jump over directly.  Function codes are found prefixing
  the panel identifiers in the upper right hand corner of each display.
```

As this panel shows, almost every Administrator Interface panel has a brief help display that you invoke using the PF1 key or the **?** function code.

The PF3 key exits a panel and returns you to the parent panel, the PF8 key browses forward through a series of panels, and the PF7 key browses backward through a series of panels.

Most of the Administrator Interface panels have a panel ID in the upper-left corner. The two-character prefix of the panel ID is the function code for the panel.

Note: You cannot jump to all panels directly. Some panels must have the displayed item selected directly or browsed from a previous panel.

Exiting the Administrator Interface

You can use the Clear key or X function code to exit the Administrator Interface from any panel. After exiting, the following message appears, confirming the exit.

ADMINISTRATOR INTERFACE TERMINATED

Bypassing the Main Menu

In order to enter the Administrator Interface faster, you can bypass the main menu by entering the transaction ID for each major component directly from a cleared screen. Once inside the Administrator Interface, you can always jump to these three major panels by entering their function code or PF key.

Trans Code	Primary Panel Name for Component	Function Code	PF Key
IPAU	Utilization Summary panel	U	PF4
IPAC	Configuration Table panel	C	PF5
IPAO	Operations Control Menu	O	None

Note: It is recommended that only these three transaction IDs and IPAM be used to directly enter the Administrator Interface. Using any of the other Administrator Interface transactions with the IPA prefix can have unpredictable results.

Utilization

The Administrator Interface's Utilization component provides a realtime display of all the CICS/TS tasks in the system that have a TCP connection or a UDP endpoint established.

Utilization panels can also display detail information about the current state of each connection endpoint and enable the administrator to perform the following types of actions on those connections:

- Terminate any active connection by making an abortive close on the connection
- Ping the remote host of a connection to make sure that the remote host is still alive
- Dynamically set and remove trace options on a connection for any Unicenter SOLVE:CPT tool or API program, and observe the trace output in the Online Trace

These features give a CICS/TS administrator the ability to monitor and diagnose all of the CICS/TS transactions that are using TCP or UDP on that CICS/TS system.

Utilization Summary Panel

The Utilization Summary panel is the heart of the Administrator Interface. It displays all of the endpoints established through the Unicenter SOLVE:CPT Interface.

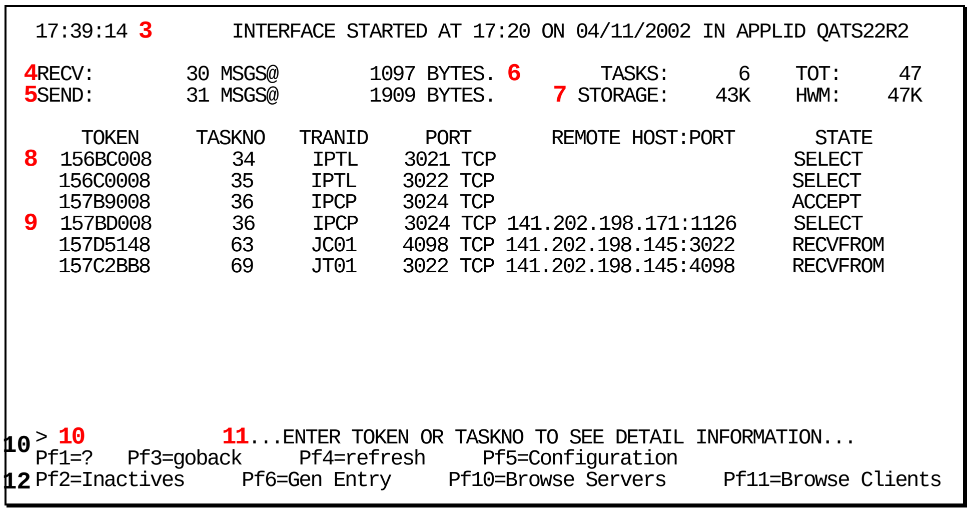
Three types of Unicenter SOLVE:CPT endpoints can be displayed:

- A TCP connection endpoint
- A UDP endpoint (which is connection-less)
- A listening endpoint (which is waiting for a connection)

Each of these endpoint types can be active and owned by a CICS/TS task, or inactive with no current task owning it.

The following is an example a Utilization Summary.

```
1 mm/dd/yyyy                    UTILIZATION SUMMARY                    2 US01
```



```
 17:39:14 3      INTERFACE STARTED AT 17:20 ON 04/11/2002 IN APPLID QATS22R2

4RECV:        30 MSGS@       1097 BYTES. 6      TASKS:      6    TOT:     47
5SEND:        31 MSGS@       1909 BYTES.    7 STORAGE:    43K    HWM:    47K

    TOKEN     TASKNO    TRANID     PORT       REMOTE HOST:PORT       STATE
8  156BC008      34      IPTL     3021 TCP                          SELECT
   156C0008      35      IPTL     3022 TCP                          SELECT
   157B9008      36      IPCP     3024 TCP                          ACCEPT
9  157BD008      36      IPCP     3024 TCP 141.202.198.171:1126     SELECT
   157D5148      63      JC01     4098 TCP 141.202.198.145:3022     RECVFROM
   157C2BB8      69      JT01     3022 TCP 141.202.198.145:4098     RECVFROM






10 > 10            11...ENTER TOKEN OR TASKNO TO SEE DETAIL INFORMATION...
 Pf1=?    Pf3=goback     Pf4=refresh     Pf5=Configuration
12 Pf2=Inactives     Pf6=Gen Entry     Pf10=Browse Servers     Pf11=Browse Clients
```

Where:

1 Current date and time on this CICS/TS system.

2 Panel title and panel ID for this screen.

3 Date and time the Unicenter SOLVE:CPT Interface was initialized, and the APPLID of this CICS/TS system.

4 Total number of Unicenter SOLVE:CPT messages received by all Unicenter SOLVE:CPT endpoints since Unicenter SOLVE:CPT initialization, and their cumulative byte length.

5 Total number of Unicenter SOLVE:CPT messages sent by all Unicenter SOLVE:CPT endpoints since Unicenter SOLVE:CPT initialization, and their cumulative byte length.

6 Current number of active Unicenter SOLVE:CPT tasks in this system, and the total number of Unicenter SOLVE:CPT tasks that have executed since Unicenter SOLVE:CPT initialization.

7 Total amount of storage currently being used for Unicenter SOLVE:CPT control blocks and buffers, and the high-water-mark for that storage since Unicenter SOLVE:CPT initialization.

8 Inactive Unicenter SOLVE:CPT endpoints that have no active CICS/TS task attached.

9 Active Unicenter SOLVE:CPT endpoints that display the TASKNO of their attached CICS/TS task, and the host IP Address and port number of their remote partner.

10 Command line used for selecting the detail display of a particular connection, by entering its TASKNO or TOKEN address.

11 Message area for command feedback and error notification.

12 PF keys currently active for this panel.

Summary Statistics

The Summary Statistics shown at the top of the Utilization Summary panel are cumulative for all Unicenter SOLVE:CPT transactions in this CICS/TS system.

- The total number of Unicenter SOLVE:CPT messages sent and received since Unicenter SOLVE:CPT initialization and their cumulative byte length are shown on the left
- The current number of active Unicenter SOLVE:CPT tasks that have executed since Unicenter SOLVE:CPT initialization are shown on the right
- Below the Unicenter SOLVE:CPT messages and tasks is the total amount of storage currently used for Unicenter SOLVE:CPT control block and buffers, and the high-water-mark (HWM) for that storage since Unicenter SOLVE:CPT initialization

Note: If the Reset All Statistics function is performed from the Operations Control menu, the message and byte counts are reset to zero, and the task and storage counts are reset to their current values.

List of Unicenter SOLVE:CPT Endpoints

Each page of the Utilization Summary can display 13 Unicenter SOLVE:CPT endpoints. The endpoint for a TCP connection can exist across CICS/TS tasks and is passed via the GIVE and TAKE API calls.

If a CICS/TS task currently owns an endpoint, then the connection is active and the task number of the owning CICS/TS task is displayed in the TASKNO column. If the endpoint has no owning CICS/TS task at the moment, it is inactive and the TASKNO column is blank.

- Active endpoints—The screen line is highlighted; the task number is shown
- Inactive endpoints—The line is not highlighted; no task number is shown

The Unicenter SOLVE:CPT endpoints are displayed in task number sequence, so the inactive endpoints are shown first, then the active endpoints are shown, with the oldest tasks first.

Endpoint Data Fields

Each Unicenter SOLVE:CPT endpoint listed on the Utilization Summary panel has these data fields displayed in each of its columns:

TOKEN—The address of the control block that represents the Connection EndPoint.

TASKNO—The task number of the owning CICS/TS task. This field is blank for inactive endpoints.

TRANID—The transaction ID of the owning CICS/TS task.

PORT—The local port number being used, and the type of protocol being used (TCP or UDP).

REMOTE HOST:PORT—The IP address and port number of the partner program on the remote host. Listener tasks have blanks in this field.

STATE—The STATE column contains the last API call made on this endpoint, and the return code of that call.

Example Syntax

API Call Return Code

XXXXX-RRR

- API Calls

  TCP API Calls: CLOSE, CONNCT, GIVE, LISTEN, RECV, SEND, TAKE.

  UDP API Calls: RCFR, SNTO.

- Return Codes:

  I/P—In progress. The call is being processed by the TCP transport provider.

  OK—A zero return code was returned for the call.

  nnn—A three-digit, decimal return code, as specified in the *Message Guide* or in a return code copy member such as T09KCRCS for COBOL.

Displaying Inactive Endpoints

To shorten the list of endpoints displayed on this panel, you can suppress the display of all inactive endpoints. The inactive endpoints are not currently owned by any CICS/TS task. The PF2 key toggles the display of these inactive endpoints on the Utilization Summary panel.

When inactive endpoints are displayed, the PF2 key is shown as:

```
PF2=Inactives off
```

When inactive endpoints are suppressed, the PF2 key is shown as:

```
PF2=Inactives
```

Note: The Inactives off option remains in effect for only one administrative session. When first displaying the Utilization Summary panel the default is always Inactives.

Explicit Selection of the Detail Display

A detail display can be selected for any of the endpoints displayed on the Utilization Summary panel.

To do this, type either the TASKNO or TOKEN field number on the command line, as the default prompt for this panel suggests:

```
...ENTER TOKEN OR TASKNO TO SEE DETAIL INFORMATION...
```

The task number is the easiest field to enter for this selection. However, the TOKEN address field must be offered as an alternative for the inactive endpoints that do not currently have an owning CICS/TS task, and thus no task number to enter.

If an invalid number is entered on the command line, then the following message appears, prompting for another TASKNO or TOKEN:

```
REQUESTED TASK NOT FOUND - PLEASE VERIFY             1234
```

Browsing Detail for Client or Server Task

These are the options for browsing through the detail displays if you do not want to select a task explicitly on the Utilization Summary panel:

PF10=Browse Servers—(Connection initiated by remote host)

PF11=Browse Clients—(Connection initiated by CICS/TS task)

A client task initiates a connection with a server on a remote host. A client task does not need to specify what local port it will be using, so the TCP provider automatically assigns its connection a port number, beginning with 4096 and going up.

A server task must always keep the same port number and should not conflict with any other server or client port. There may be a listener task and several subordinate receiver tasks started by the listener that are all associated with the same server port.

Generating an Online Trace Entry

You can specify online tracing options from the Browse Detail panel, or by using PF6 on the Utilization Summary panel, as shown by this option:

```
PF6=Gen Entry
```

Use this option when you want to trace an endpoint that is not currently established. The Browse Detail panel can only specify tracing options for connection endpoints that currently exist. However, some connections complete so quickly, that the Online Trace entry must be defined before the connection initiation in order to capture it. How that trace entry is specified, and how its trace options are selected, are described further in Utilization.

Client/Server Browse Detail Panel

The Browse Detail panel displays statistics for a particular endpoint and its CICS/TS transaction. From this panel, three potential actions can be taken on the endpoint:

- End its CICS/TS task, and close the endpoint
- Ping the remote host for the endpoint to test if it is still alive
- Set API trace options for the endpoint

The Browse Detail panel can display or suppress different data fields, depending on the type of CICS/TS task and the state of the endpoint. Three examples follow that show these different types of Browse Detail displays.

Examples

This section presents three usage examples.

Example 1: Displaying an Active Connection Endpoint

The first example is a server transaction waiting to receive more input on a TCP connection with a remote client. The identification line in the center of the screen (item 6) displays the data fields that identify this endpoint:

- Task Number
- Local Port
- Token Address
- Protocol
- State (last API call, and Return Code)

Note: This identification is very similar to the line on the Utilization Summary for this endpoint. Refer to Utilization for detailed descriptions of these fields.

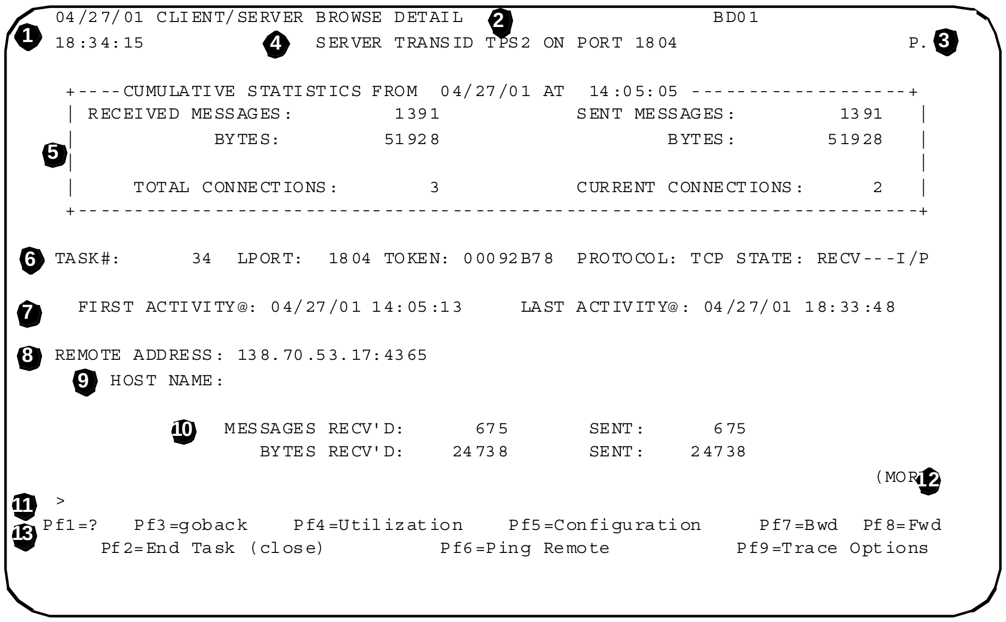
The First Activity and Last Activity fields show the date and time of the first API call made on this connection, and the data and time of the most recent API call made, which is shown in the State field along with its return code. In this case, the I/P in the State field indicates that the Receive call is still 'In Progress'. The Server task is waiting for its last call to receive more input on the connection.

Because this endpoint has an active connection with a remote client, all the remote host information is shown, and the Endpoint Statistics are shown.

The remote host information shows the IP address of the remote host in dotted decimal notation. Following the colon is the port number being used by the Client program on the remote host. If the Domain Name Resolver is active, then the domain name of the remote host will also be displayed.

Server Transaction Detail

The following figure shows a page of the Browse Detail panels invoked by the Browse Servers function. It displays one of the two current server transactions receiving input on port 1804.



```
 04/27/01 CLIENT/SERVER BROWSE DETAIL  2                  BD01
1 18:34:15           4  SERVER TRANSID TPS2 ON PORT 1804                    P.3

  +----CUMULATIVE STATISTICS FROM  04/27/01 AT  14:05:05 -------------------+
  | RECEIVED MESSAGES:         1391            SENT MESSAGES:          1391   |
  |            BYTES:         51928                    BYTES:         51928   |
 5|                                                                           |
  |     TOTAL CONNECTIONS:        3            CURRENT CONNECTIONS:      2    |
  +---------------------------------------------------------------------------+

6 TASK#:       34  LPORT:  1804 TOKEN: 00092B78  PROTOCOL: TCP STATE: RECV---I/P

7   FIRST ACTIVITY@: 04/27/01 14:05:13      LAST ACTIVITY@: 04/27/01 18:33:48

8 REMOTE ADDRESS: 138.70.53.17:4365
   9 HOST NAME:

           10  MESSAGES RECV'D:       675       SENT:       675
                  BYTES RECV'D:     24738       SENT:     24738
                                                                      (MOR12
11 >
13 Pf1=?   Pf3=goback    Pf4=Utilization    Pf5=Configuration     Pf7=Bwd   Pf8=Fwd
      Pf2=End Task (close)          Pf6=Ping Remote           Pf9=Trace Options
```

Where:

1 Current date and time on this CICS/TS system.

2 Title of this panel.

3 Panel ID and current page number.

4 Transaction ID of the server task and the local port it is using.

5 Cumulative statistics being collected for all instances of the server transactions using this port number.

6 Identification line for this TCP endpoint, giving its task number, local port number, token address, protocol, and current state.

7 Date and time of the first API call on this connection, and the date and time of the most recent API call, which is shown in the state field.

8 IP address of the remote host in dotted decimal notation, followed by the port number being used by the client program on the remote host.

9 Domain name of the remote host as resolved by the DNR subsystem.

10 Endpoint statistics collected for just this specific connection.

11 Command line used for specifying an Administrator Interface function code.

12 (MORE) indicator that more panels are available as part of a browse.

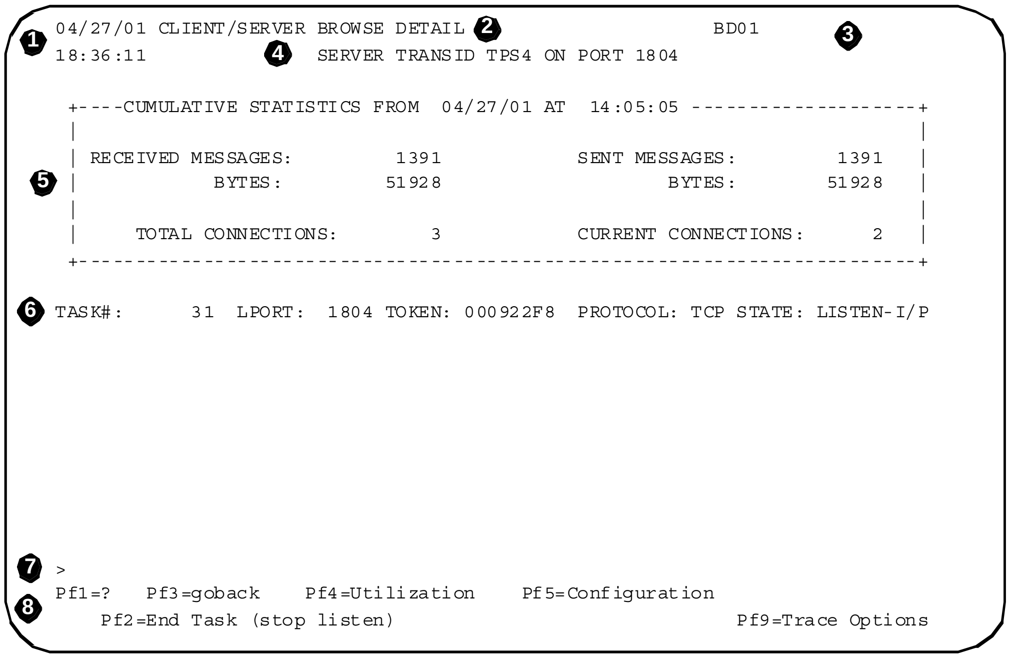
13 PF keys currently active for this panel.

The counters for both the Cumulative Statistics and the Endpoint Statistics can be reset to zero from the Operations Menu in the Administrator Interface, as described in Operations. The date and time shown in the Cumulative Statistics box is the last time the statistics were reset or when the Unicenter SOLVE:CPT Interface was initialized.

Example: 2
Listener Transaction Detail

This second example is a Unicenter SOLVE:CPT listener task waiting to receive new connection requests on the same local port number as the previous server example.

The following figure shows a single Browse Detail panel that was selected directly by TASKNO from the Utilization Summary panel.



```
1 04/27/01 CLIENT/SERVER BROWSE DETAIL 2                  BD01       3
  18:36:11            4  SERVER TRANSID TPS4 ON PORT 1804

   +----CUMULATIVE STATISTICS FROM  04/27/01 AT  14:05:05 -------------------+
   |                                                                         |
   | RECEIVED MESSAGES:          1391             SENT MESSAGES:         1391   |
 5 |             BYTES:         51928                     BYTES:        51928   |
   |                                                                         |
   |     TOTAL CONNECTIONS:        3             CURRENT CONNECTIONS:      2   |
   +-------------------------------------------------------------------------+

6 TASK#:      31  LPORT:  1804 TOKEN: 000922F8  PROTOCOL: TCP STATE: LISTEN-I/P




7 >
  Pf1=?   Pf3=goback    Pf4=Utilization    Pf5=Configuration
8     Pf2=End Task (stop listen)                                 Pf9=Trace Options
```

Where:

1 Current date and time on this CICS/TS system.

2 Title of this panel.

3 Panel ID.

4 Transaction ID of the server task and the local port it is using.

5 Cumulative statistics being collected for all instances of the server transactions using this port number.

6 Identification line for this listen endpoint, giving its task number, local port number, token address, protocol, and current state.

7 Command line used for specifying an Administrator Interface function code.

8 PF keys currently active for this panel.

Note: This panel is missing the following data fields that were included in Example 1 (Browse Detail) :

- No remote host information is shown because a listener task has no established connection. It is just waiting for a new connection request to come in from any remote host.
- No endpoint statistics are shown because the listener task has only a listening endpoint, and no established connection.
- No PF6 key to ping the remote host since no connection exists.
- No PF7 or PF8 key since this panel is not part of a browse.
- No page number is shown since this panel is not part of a browse.

Example 3
Displaying a Listener Endpoint

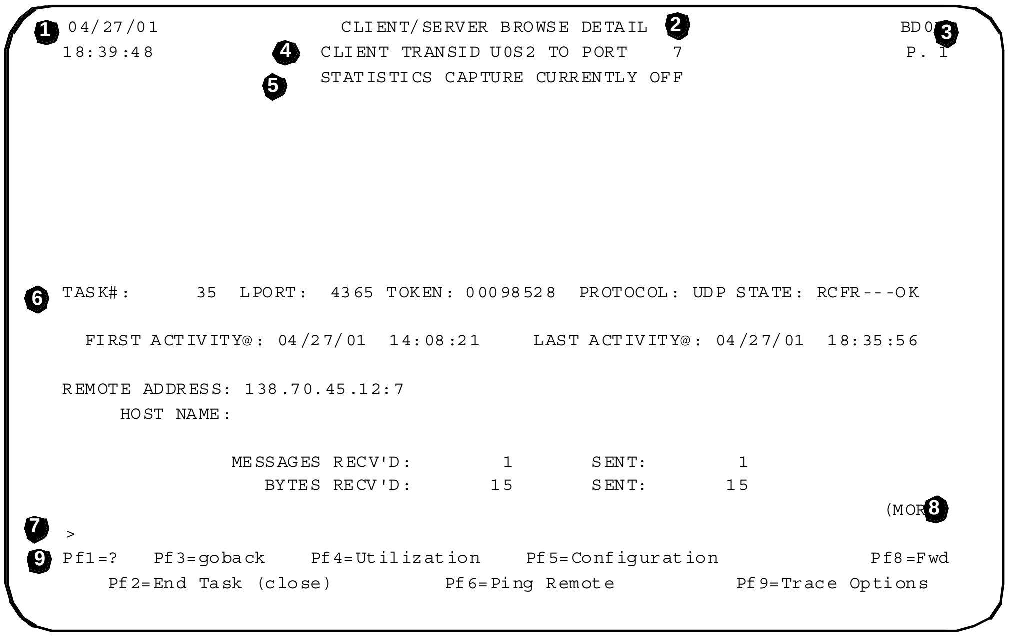
A listener task has an endpoint, but no connection with a remote host. It usually starts a separate server task when a new connection request comes in from a remote host. This is why the bottom half of the screen is empty for a listener task, but is filled in for its server task.

The Cumulative Statistics shown in this example are the same as those shown in the server task example, because the counts for connections and data transmissions are accumulated based on the local port number of the servers. The listener task and all of its subordinate server tasks (which it started) all have the same local port number, and will all show the same set of Cumulative Statistics.

Example: 4
Client UDP
Transaction Detail

This example is a UDP Client transaction that sends a datagram to the echo port of a remote host.

The following figure shows a page of the Browse Detail panels invoked by the Browse Clients function.



```
 04/27/01                    CLIENT/SERVER BROWSE DETAIL                       BD0
18:39:48                  CLIENT TRANSID U0S2 TO PORT    7                     P. 1
                          STATISTICS CAPTURE CURRENTLY OFF







TASK#:      35  LPORT:  4365 TOKEN: 00098528  PROTOCOL: UDP STATE: RCFR---OK

  FIRST ACTIVITY@: 04/27/01  14:08:21    LAST ACTIVITY@: 04/27/01  18:35:56

REMOTE ADDRESS: 138.70.45.12:7
     HOST NAME:

               MESSAGES RECV'D:        1       SENT:        1
                  BYTES RECV'D:       15       SENT:       15
                                                                      (MOR
 >
Pf1=?   Pf3=goback    Pf4=Utilization    Pf5=Configuration             Pf8=Fwd
    Pf2=End Task (close)          Pf6=Ping Remote          Pf9=Trace Options
```

Displaying a UDP
Endpoint

A UDP task has an endpoint, but no connection with a remote host, whether it is a client or a server transaction.

Because a UDP client must specify a remote host and port number, that information can be displayed, along with the endpoint statistics for amount of data sent and received by the UDP task.

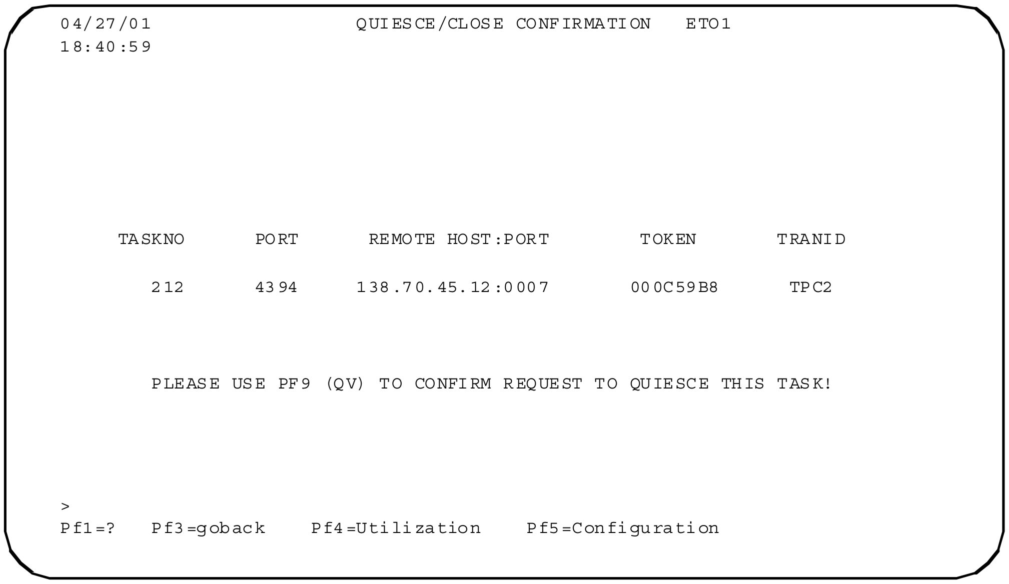
Note: The top half of the screen is empty where the Cumulative Statistics would appear. This is not because the Client is a UDP task, but because AISTATOP=NO was specified in the Unicenter SOLVE:CPT Configuration Table for this execution of CICS/TS. When this parameter is set to NO, every Browse Detail panel displays the following message:

STATISTICS CAPTURE CURRENTLY OFF.

Terminating the Endpoint Task

Use the PF2 key to terminate the CICS/TS task for the endpoint displayed on the Browse Detail panel. This termination is done by an abortive close on the Unicenter SOLVE:CPT endpoint, not by abending the CICS/TS task explicitly.

This action causes a disconnect return code to any outstanding API call from the CICS/TS task, which must then interpret that disconnect return code, and terminate via its own logic. When the PF2 key is pressed, the Quiesce/Confirm panel appears to confirm the close of the displayed task.



```
04/27/01                 QUIESCE/CLOSE CONFIRMATION   ETO1
18:40:59







       TASKNO      PORT      REMOTE HOST:PORT        TOKEN       TRANID

          212      4394     138.70.45.12:0007       000C59B8       TPC2



          PLEASE USE PF9 (QV) TO CONFIRM REQUEST TO QUIESCE THIS TASK!




>
Pf1=?   Pf3=goback    Pf4=Utilization    Pf5=Configuration
```

Confirming the Quiesce

The PF9 key confirms the termination, or the QV function code can be entered on the command line.

After confirmation, the next panel that appears is the Utilization Summary panel, to verify the status of the task.

The following response message appears in the message area of that panel:

```
>  .....REQUESTED CONNECTION(S) CLOSED .....
```

If the termination is to be aborted by not confirming on the Quiesce/Confirm panel, any of the standard navigation function codes or PF keys can be used to go to another panel. The PF3 key returns to the Utilization Summary panel, not the previously displayed Browse Detail panel.

Pinging the Remote Host

Use the PF6 key to send a ping to the remote host displayed on the Browse Detail panel.

The ping is an ICMP echo request that does not use the displayed connection, but can test if the remote host is still reachable and if its system is still active. If a break has occurred in the route to the host, or if its system has crashed, the following message appears on the Browse Detail panel:

```
> ....NO RESPONSE FROM REMOTE HOST....
```

If the ping is successful, then the following message appears:

```
> ....REMOTE HOST IS ALIVE....
```

This feature can help in determining if the displayed endpoint needs to be terminated because the remote host is no longer reachable.

Configuration

The Configuration component of the Administrator Interface enables you to display and modify the Unicenter SOLVE:CPT tool definitions made in the Unicenter SOLVE:CPT configuration table. The Administrator Interface now lets you modify many of the definitions of the Unicenter SOLVE:CPT tools in the following ways:

- Update the parameters of any Listen/Receive tool
- Add a new Listen/Receive tool to the configuration table
- Start any of the Listen/Receive tools defined to the configuration table

Some global parameters of the configuration table are also displayed by the Configuration component, but they cannot be modified. This is because they define the attributes of the currently running Unicenter SOLVE:CPT Interface (such as the subsystem ID of its TCP transport provider), which cannot be modified without stopping and restarting the Unicenter SOLVE:CPT Interface.

Note: The *dynamic updates* to the tool parameters, and the addition of new tool definitions, are *not permanent changes* to the configuration table. The next time the Unicenter SOLVE:CPT Interface is initialized, the configuration tools revert to those definitions assembled into load module T09CONez, depending on the suffix identified.

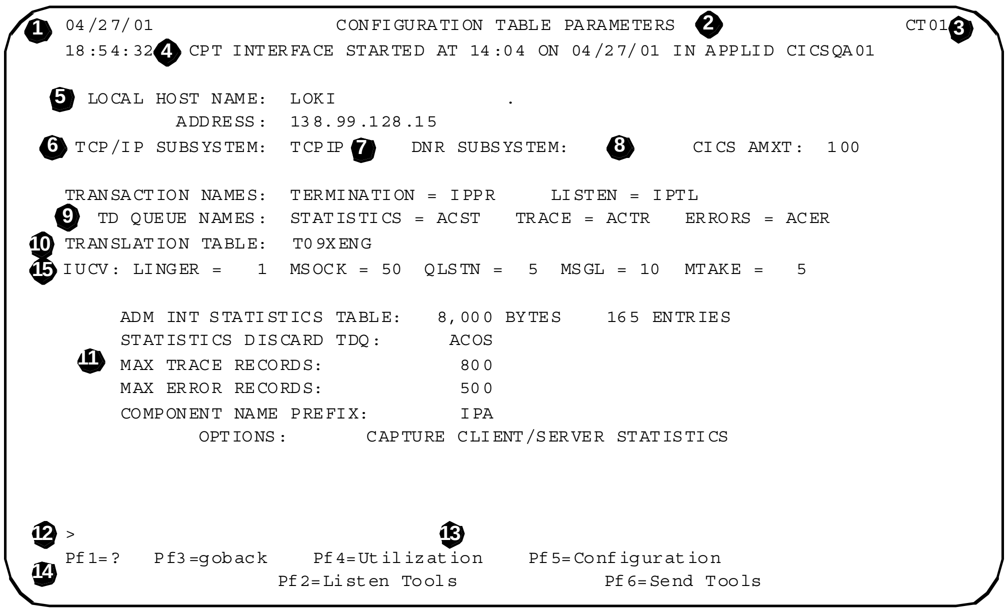
Configuration Table Parameters Panel

The Configuration Table Parameters panel displays all of the global attributes of the Unicenter SOLVE:CPT Interface that is currently running. These attributes are specified by the T09MCICS macro coded at the start of the configuration table. This panel shows both the default attributes, and those that were set by macro parameters for the current definition of the Unicenter SOLVE:CPT Interface. These parameters cannot be modified dynamically.

The Configuration Table Parameters panel is invoked by the C function code from anywhere in the Administrator Interface, or by using the PF5 key, when this option is displayed at the bottom of a panel:

`PF5=Configuation`

The following is a sample of the Configuration Table Parameters panel.



```
04/27/01                    CONFIGURATION TABLE PARAMETERS                    CT01
18:54:32  CPT INTERFACE STARTED AT 14:04 ON 04/27/01 IN APPLID CICSQA01

  LOCAL HOST NAME:  LOKI                    .
          ADDRESS:  138.99.128.15
 TCP/IP SUBSYSTEM:  TCPIP     DNR SUBSYSTEM:          CICS AMXT:  100

TRANSACTION NAMES:  TERMINATION = IPPR     LISTEN = IPTL
   TD QUEUE NAMES:  STATISTICS = ACST   TRACE = ACTR   ERRORS = ACER
TRANSLATION TABLE:  T09XENG
IUCV: LINGER =   1  MSOCK = 50  QLSTN =  5  MSGL = 10  MTAKE =   5

     ADM INT STATISTICS TABLE:   8,000 BYTES    165 ENTRIES
     STATISTICS DISCARD TDQ:      ACOS
     MAX TRACE RECORDS:            800
     MAX ERROR RECORDS:            500
     COMPONENT NAME PREFIX:        IPA
            OPTIONS:       CAPTURE CLIENT/SERVER STATISTICS



>
Pf1=?   Pf3=goback    Pf4=Utilization    Pf5=Configuration
                  Pf2=Listen Tools             Pf6=Send Tools
```

Where:

1 Current date and time on this CICS/TS system.

2 Title of this panel.

3 Panel ID.

4 The date and time the Unicenter SOLVE:CPT Interface was initialized and the APPLID of this CICS/TS system.

5 The local host name and IP address of the TCP transport provider to which this Unicenter SOLVE:CPT Interface is connected.

6 The subsystem ID by which the TCP transport is specified.

7 The subsystem ID by which the Domain Name Resolver for IP address/name translations. (Only for TCPaccess.)

8 The CICS/TS AMXT parameter from the SIT table specifying Active Max Tasks.

9 The transaction names and transient data queue names used by Unicenter SOLVE:CPT.

10 The load module containing the ASCII/EBCDIC translation table.

11 The parameters for the Administrator Interface itself.

12 The command line used for entering a function code.

13 The message area for command feedback and error notification.

14 The PF keys currently active for this panel.

15 Only displays when running with IBM TCP/IP.

Configuration Table Description

Most of these attributes will rarely change for the Unicenter SOLVE:CPT Interface. The following list provides a quick overview of the attributes before browsing the Unicenter SOLVE:CPT tool definitions.

Local Host Name	Full domain name of the local host's TCP/IP subsystem.
Address	Dotted decimal IP address of the local host's TCP/IP subsystem.
TCP/IP Subsystem	Subsystem ID of the TCP/IP transport provider.

Note: The local host name and address are not explicitly specified in the configuration table. They are derived from the Jobaname= parameter that specifies which subsystem will provide TCP/IP services for this CICS/TS.

DNR Subsystem	Subsystem ID of the Domain Name Resolver for IP Name/IP Address translation.
CICS AMXT	CICS/TS Active Max Task value from the CICS/TS system initialization table.

Note: This CICS/TS parameter is shown here because Listener tasks are long running tasks. Creating and starting new Listener tools could adversely impact a CICS/TS system with a low AMXT value.

Transaction Names	The configuration table lets you change these transids, if the default names are already in use by this CICS/TS.
TD Queue Names	The configuration table lets you change these queue names, if the default names are already in use by this CICS/TS.

Translation Table	The configuration table lets you specify a custom ASCII/EBCDIC translation table, if the default table provided is not sufficient. **Note:** The Configuration Table Parameters panel shows the default values for all the transaction names, queue names, and translation table.
Linger	Shows the default number of seconds to wait after an orderly close call.
MSOCK	Shows the default maximum number of sockets allowed for the IUCV path of each Listener task on a port.
QLSTN	Default number of connection requests that can be queued in a backlog for each Listener task on a port.
MSGL	Maximum number of IUCV messages allowed at any one time.
MTAKE	Maximum number of seconds to wait for a TAKE call to be issued for a token. The task started by the Listener should issue this TAKE call after it retrieves the token passed to it by the Listener.
Adm Int Statistics Table	Number of bytes and the number of entries allocated to the internal statistics table used by the Administrator Interface. Default: 1000 bytes. **Note:** The Configuration Table Parameters panel shows the values when parameter AITABSZ=8000 is specified in the configuration table.
Statistics Discard Queue	When the Cumulative Statistics are reset, or if Unicenter SOLVE:CPT is shut down, the entries in the statistics table are written as data records to this transient data queue. Default: No queue defined. **Note:** If the queue name specified in this field is not defined in the CICS/TS RDO, then the following error message appears: TRANSIENT DATA QUEUE *XXXX* IS NOT DEFINED TO CICS
Max Trace Records	Maximum trace entries that can be collected by the Online Trace. You can reset this value from the Operations Control menu. Default: Zero.
Max Error Records	Maximum Error Log record entries that can be displayed from the Operations Control menu. Default: 30.

Component Name Prefix	Display the three-character prefix of all the transids used for the Administrator Interface. Default: IPA.
Options	Appears when Cumulative Statistics are being collected for Unicenter SOLVE:CPT transactions. **Note:** The Configuration Table Parameters panel shows the option line displayed. If AITSTATOP=YES is not specified in the configuration table, this line does not appear.

At the bottom of this panel are the PF key selections for browsing the Unicenter SOLVE:CPT tools defined by this configuration table. PF2 starts the browse of the Listen/Receive tools, and PF6 starts the browse of the Send tools.

Operations

The Operations component of the Administrator Interface provides a collection of functions to diagnose and manage Unicenter SOLVE:CPT. The Unicenter SOLVE:CPT Task-Related User Exit (TRU Exit) provides the call interface to the TCP transport provider address space. This Task-Related User Exit must be enabled to CICS/TS before any transaction can execute a Unicenter SOLVE:CPT call, and when it is disabled to CICS/TS, all TCP and UDP endpoints are terminated.

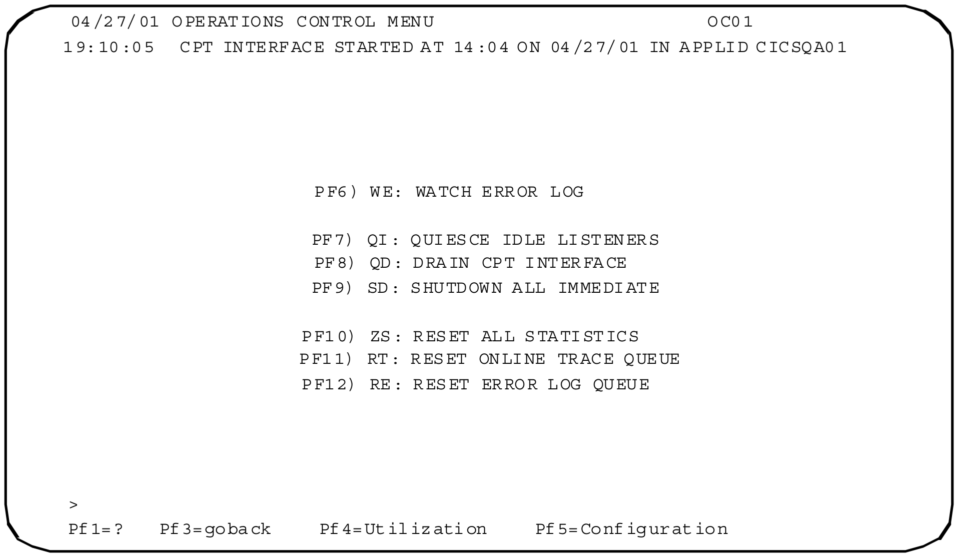
This Task-Related User Exit is referred to in this section as the *Unicenter SOLVE:CPT Interface*. The process of enabling the TRU Exit is referred to as *Unicenter SOLVE:CPT startup*, and the process of disabling the TRU Exit is referred to as *Unicenter SOLVE:CPT shutdown*.

Operations Control Menu

The Operations Control Menu provides functions to diagnose and manage the Unicenter SOLVE:CPT Interface. These functions are summarized first for the menu, and then the panels they invoke are described.

This Operations Control Menu is invoked by the O function code from anywhere within the Administrator Interface.

The following is an example of the Operations Control Menu.



```
 04/27/01 OPERATIONS CONTROL MENU                            OC01
19:10:05  CPT INTERFACE STARTED AT 14:04 ON 04/27/01 IN APPLID CICSQA01




                        PF6) WE: WATCH ERROR LOG

                        PF7) QI: QUIESCE IDLE LISTENERS
                        PF8) QD: DRAIN CPT INTERFACE
                        PF9) SD: SHUTDOWN ALL IMMEDIATE

                       PF10) ZS: RESET ALL STATISTICS
                       PF11) RT: RESET ONLINE TRACE QUEUE
                       PF12) RE: RESET ERROR LOG QUEUE




 >
 Pf1=?   Pf3=goback    Pf4=Utilization    Pf5=Configuration
```

Operations Functions

The following operation functions can be selected using the PF key shown before it on each line of the menu, or they can be selected by entering the two-character function code shown after the PF key. These function codes can also be entered on other Administrator Interface panels to jump to these operation functions directly without going through this menu panel.

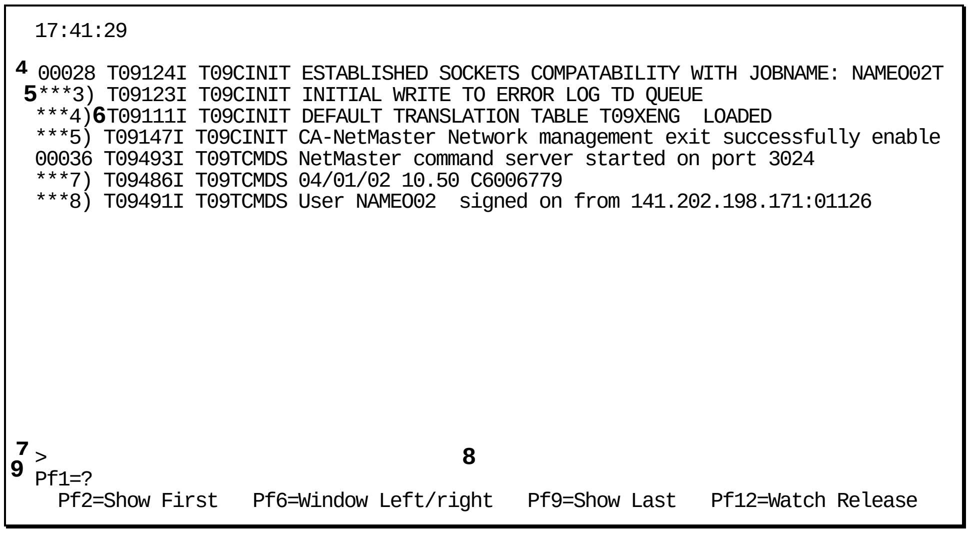
Watch Error Log	Lets the administrator browse the Unicenter SOLVE:CPT error messages that are normally sent to the ACER Transient Data queue. These messages are displayed by a wrap-around queue of Temp Storage records similar to the Online Trace queue.
Quiesce Idle Listeners	Lets the administrator close the endpoints of all currently idle Listener tasks. An idle Listener task is one that currently has no active connections established to its local port number.
Drain Unicenter SOLVE:CPT Interface	Lets the administrator close the endpoints of all Listener tasks to prevent a new connection being accepted from a remote host, and it prevents any CICS/TS task from making a connect call to a remote host. Draining the Interface implies that all existing connections are allowed to complete, but establishing a new connection is blocked, in preparation for Unicenter SOLVE:CPT shutdown.
Shutdown All Immediate	Lets the administrator force an abortive close on all active connections, and invoke the Unicenter SOLVE:CPT termination routine, just the same as the IPPR transaction. This action shuts down the interface immediately.
Reset All Statistics	Lets the administrator reset both the cumulative statistics and the endpoint statistics that are shown on the Browse Detail and Utilization Summary panels.

Unicenter SOLVE:CPT Interface Error Log Panel

The Interface Error Log panel displays the Unicenter SOLVE:CPT error messages normally sent to the ACER Transient Data queue, which is the Unicenter SOLVE:CPT Error Log. These error messages are copied into a temporary storage queue where they can be browsed online. This enables an administrator to diagnose Unicenter SOLVE:CPT error conditions more rapidly by viewing online the messages produced by the Interface.

The following is an example of the Unicenter SOLVE:CPT Interface Error Log panel.

```
1 mm/dd/yyyy                    INTERFACE ERROR LOG  2                 3 WE01
```



```
 17:41:29

4 00028 T09124I T09CINIT ESTABLISHED SOCKETS COMPATABILITY WITH JOBNAME: NAME002T
 5***3) T09123I T09CINIT INITIAL WRITE TO ERROR LOG TD QUEUE
  ***4)6T09111I T09CINIT DEFAULT TRANSLATION TABLE T09XENG  LOADED
  ***5) T09147I T09CINIT CA-NetMaster Network management exit successfully enable
  00036 T09493I T09TCMDS NetMaster command server started on port 3024
  ***7) T09486I T09TCMDS 04/01/02 10.50 C6006779
  ***8) T09491I T09TCMDS User NAME002  signed on from 141.202.198.171:01126









7 >                                    8
9 Pf1=?
    Pf2=Show First   Pf6=Window Left/right   Pf9=Show Last   Pf12=Watch Release
```

Where:

1 Current date and time on this CICS/TS system.

2 Title of this panel.

3 Panel ID and page number.

4 CICS/TS task number of the endpoint task generating the traces messages.

5 A line count of subsequent messages produced by the same task.

6 Unicenter SOLVE:CPT message number for each new trace message.

8 The command line used for specifying a function code.

10 The message area for command feedback and error notification.

11 The PF keys currently active for this panel.

Error Log Queue Description

The Error Log queue is a Temp Storage queue whose size is determined by the AISELOG= parameter of the T09MCICS macro specified in the configuration table. The default size of the Error Log queue is 30 records, and its maximum size is 999 records. When the Error Log queue becomes full, the error messages wrap around and start overlaying the first messages in the queue.

This queue can be changed in size or cleared by the RE (Reset Error) function on the Operations Control Menu panel. If the Error Log queue is set to zero, the following error message appears on the Unicenter SOLVE:CPT Interface Error Log panel:

```
...NO ERROR RECORDS LOGGED...
```

Note: The Error Log queue contains only records from the ACER Transient Data destination. Statistics records sent to ACST and Trace records sent to ACTR destinations are not displayed here, even though these records may be mixed together in a CICS/TS sysout data set.

Browsing the Error Log Queue

PF key selections let the user browse forward and backward through the messages, scroll the display left and right, and jump to the first or last message in the queue. When finished with browsing the error messages, you can use the PF12 key to exit the Unicenter SOLVE:CPT Interface Error Log panel.
The PF12 key releases the Error Log queue, and goes to the Utilization Summary panel.

When more than one user is browsing the Error Log queue, only the first user has the PF12 option to release the Error Log queue. The other users will have to use the O function code to return to the Operations Control menu.

Note: New error messages produced while the Unicenter SOLVE:CPT Interface Error Log panel is displayed cannot be added to the Error Log queue. In order to see if any new error messages have arrived, you must use PF12 to release the Error Log queue and exit the panel. Then the WE function can display the Unicenter SOLVE:CPT Interface Error Log panel again.

If the CICS/TS system is brought down and then warm-started, the Temp Storage queue containing the Error Log messages may be kept. The administrator should be aware that error messages displayed on the Unicenter SOLVE:CPT Interface Error Log panel could be from a previous execution of the CICS/TS system, if the Error Log queue was not reset with the RE function.

Quiesce Idle Listeners Function

Entering the QI function code, or using PF7 on the Operations Control Menu panel causes the Quiesce/Close Confirmation panel to appear, prompting for a confirmation of the request with this message:

```
PLEASE USE PF9 (QV) TO CONFIRM REQUEST TO QUIESCE IDLE TASKS!
```

By using the PF9 key or entering the QV function code, the administrator confirms that all idle Listeners tasks will have their endpoints closed and tasks terminated. An idle Listener task is one that currently has no active connections established to its local port number.

If the quiesce request is to be aborted by not confirming on the Quiesce/Close Confirmation panel, you can use PF3 to return to the Operations Control Menu, or you can use any of the standard function codes or PF keys to go to another panel.

Drain Unicenter SOLVE:CPT Interface Function

Entering the QD function code, or using PF8 on the Operations Control Menu causes the Quiesce/Close Confirmation panel to appear, prompting for a confirmation of the request with the following message:

```
PLEASE USE PF9 (QV) TO CONFIRM REQUEST TO DRAIN THE INTERFACE!
```

By using PF9 key or entering the QV function code, the administrator confirms that no new connections can be started by local CICS/TS tasks or from clients on remote hosts. This function causes all Listener tasks to have their endpoints closed and tasks terminated, in preparation for terminating the Unicenter SOLVE:CPT Interface.

If the drain request is aborted by not confirming on the Quiesce/Close Confirmation panel, you can use PF3 to return to the Operations Control Menu, or you can use any of the standard function codes or PF keys to go to another panel.

Shutdown All Immediate Function

Entering the SD function code, or using PF9 on the Operations Control Menu panel causes a single prompt to display on the Operations Control Menu, prompting for a verification of the shutdown with the following message:

```
PLEASE USE PF9 (SV) TO CONFIRM COMPLETE SHUTDOWN REQUEST!
```

By using the PF9 key or entering the SV function code, the administrator confirms that the Unicenter SOLVE:CPT Interface should be terminated and that any remaining connection endpoints will have an abortive close. After entering the confirmation, the next message displayed should report the termination of the Unicenter SOLVE:CPT Interface:

```
T09120I  T09TTERM TERMINATION SUCCESSFUL
```

If the shutdown request is aborted by not confirming on the Operations Control Menu, you can use PF3 to return to the Main Menu panel, or you can use any of the standard function codes or PF keys to go to another panel.

Reset All Statistics Function

Entering the ZS function code, or using PF10 on the Operations Control Menu causes the Reset Statistics panel to appear, prompting for a confirmation of the request with this message:

```
PLEASE USE PF9 (ZV) TO CONFIRM REQUEST TO RESET STATISTICS
```

By using the PF9 key or entering the ZV function code, the administrator confirms that both the Cumulative Statistics and the Endpoint Statistics being collected for all endpoints will be cleared.

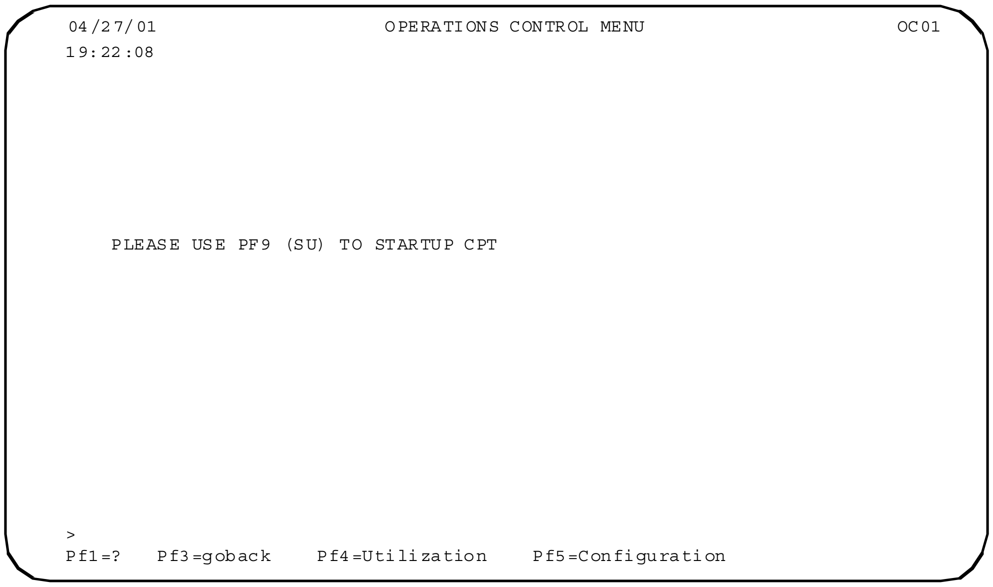
If the reset request is aborted by not confirming on the Reset Statistics panel, you can use PF3 to return to the Operations Control Menu panel, or you can use any of the standard function codes or PF keys to go to another panel.

Startup Unicenter SOLVE:CPT Interface Function

The Startup function for Unicenter SOLVE:CPT is not normally displayed on the Operations Control Menu, because the Unicenter SOLVE:CPT Interface is usually active when using it. However, if the Unicenter SOLVE:CPT Interface was terminated or did not initialize successfully, this message displays on the Main Menu for the Administrator Interface:

```
T09122E T09AUTIL INTERFACE NOT AVAILABLE (CICS 'AEY9')
```

If the O function code or PF6 is used on the Main Menu at that time, a special version of the Operations Control Menu appears to start up the Unicenter SOLVE:CPT Interface, as shown below.



```
04/27/01                    OPERATIONS CONTROL MENU                    OC01
19:22:08



    PLEASE USE PF9 (SU) TO STARTUP CPT








>
Pf1=?   Pf3=goback   Pf4=Utilization   Pf5=Configuration
```

This version of the Operations Control Menu has only one function to confirm the startup of the Unicenter SOLVE:CPT Interface. By using the PF9 key or entering the SU function code, the administrator confirms that the Unicenter SOLVE:CPT Interface should be initialized. This action is equivalent to using the IPST transaction to start the Unicenter SOLVE:CPT Interface.

The next panel to appear is the Utilization Summary panel. The administrator can just use the Enter key on the Utilization Summary panel to see if the Listener tasks defined in the configuration table have started. Then you can use the WE function to go to the Unicenter SOLVE:CPT Interface Error Log panel to check the Unicenter SOLVE:CPT initialization messages that are written to the Unicenter SOLVE:CPT Error Log.

Index