CHAPTER 1

Assembler Language Macro Instructions

This chapter provides detailed coding information for the Cisco IOS for S/390 API assembler macro instructions. It includes these sections:

- Conventions, Definitions, and Terminology

  Describes the conventions and terminology used to describe the macro instructions, and defines the basic forms and formats that apply. Also defines register usage for the standard API calling sequences.

- Description of Macro Instructions

  Provides detailed descriptions of all API macro instructions.

Conventions, Definitions, and Terminology

Describes the conventions and terminology used to describe the macro instructions, and defines the basic forms and formats that apply. Also defines register usage for the standard API calling sequences.

Basic Format of Macro Descriptions

Each macro instruction is described using this documentation format:

- A brief introductory statement summarizing the primary use of the macro instruction
- A detailed assembler format description
- A detailed description of each macro instruction operand
- A detailed assembler format description, and a detailed description of each macro instruction operand, including completion information, a list of return codes presented in tabular format, and general usage guidelines. The basic components of each macro instruction description are described in the following topics.

Assembler Format Description

The assembler format description is a three-column table that shows how the macro instruction is to be coded. Because macro instructions are coded in the same format as assembler instructions, the three columns correspond to an assembler instruction's name, operation, and operand fields.

This is how these fields are used:

Name

The macro instruction name provides a label for the macro instruction. The name, if used, can be specified as any symbolic name valid in the assembler language.

Operation

This field contains the mnemonic operation code of the macro instruction. It is always coded exactly as shown.

Operands

The operands provide information that is required for execution of the macro instruction. Generally, this information is organized into a parameter list by the macro instruction expansion, and provided to the API during program execution. All of the macro instruction's operands are indicated in the operands column of the assembler format description.

The notation used for describing macro instruction operands is similar to that used by IBM to describe macro instructions incorporated within their products (for example, VTAM macro instructions or MVS supervisor macro instructions). This notation is briefly defined in Macro Instruction Operand Notation.

Syntax Description

The syntax includes the assembler format, operand name, and description. Every operand description begins with a brief explanation of the operand's function. If the operand has more than one fixed value that can be supplied with it, the operand description also explains the effect that each value has on the action performed by the macro instruction.

For TPL-based (Transport Service Parameter List) macro instructions, the same operand can be coded on many different macro instructions. In some cases, the coding rules and interpretation of the operand are identical. In other cases, the coding rules may differ and the interpretation may depend on the context in which it is used. To avoid unnecessary repetition, the operands that apply to several macro instructions are documented in full only for the TPL macro instruction. For other macro instructions, either an abbreviated description is given, or only the characteristics that apply to the particular macro instruction are described.

The operand description may include a description of the format in which the operand should be coded. This description is provided when the format is an exception to these general rules:

- When a quantity is indicated, such as a length or integer value, it can be coded as any non-relocatable expression that is valid for an A-type address constant or a Load Address (LA) assembler instruction, or the number of the register (enclosed in parentheses) that contains the value when the macro instruction is executed. Register notation is restricted to registers 2-12 when specifying a quantity.
- When an address is indicated, it can be coded as any relocatable expression that is valid for an A-type address constant or a Load Address (LA) assembler instruction, or the number of the register (enclosed in parentheses) that contains the address when the macro instruction is executed. Register notation is restricted to registers 1-12 when specifying the address of a TPL, and registers 2-12 for all other address operands.

Note The assembler instructions generated to process operands in the macro instruction expansion vary depending on the macro form used. This may have an effect on the maximum values that can be specified in certain macro instruction operands. For more information, refer to the discussion of list, generate, modify and execute forms of macro instructions in List, Generate, Modify, and Execute Forms.

Completion Information

All of the executable macro instructions pass return codes in registers, and most indicate status information in various control block fields when they are posted complete. Some macro instructions also return results in control block fields or storage areas provided by the application program. A description of the status information and data returned by the macro instruction follows the operand descriptions. This description is in terms of the conditions that exist on return to the application program after execution of the macro instruction in synchronous mode, or after execution of a TCHECK macro instruction when operating in asynchronous mode.

Return Codes

A list of all return code values generated by the macro instruction is presented in tabular form. The list is organized to show the conditional completion codes returned for normally completing macro instructions, and the combinations of recovery action codes and specific error codes returned for abnormally completing macro instructions.

All return codes are listed using their symbolic name. The *Cisco IOS for S/390 Unprefixed Messages and Codes* manual contains detailed information on each return code, including its symbolic name, the corresponding decimal and hexadecimal values, and a detailed explanation of its use and meaning.

Usage Information

The usage information describe the macro instruction's function and use, including any important rules that may apply to when and how it may be used. Special usage notes that may not have been covered in other sections of the macro instruction description are listed here.

Macro Instruction Operand Notation

The notational scheme used in the operands column of the assembler format description is similar to that used by IBM. This notation shows how, when, and where operands can be coded. These notational symbols are never coded in the macro instruction:

- Uppercase characters must be coded exactly as shown in the operands column. Lowercase italicized characters or words represent variables, or values that the application programmer must provide.
- A vertical bar (|) is the symbol for "exclusive or".

  SYNC | ASYNC means that either SYNC or ASYNC (but not both) should be coded.
- An underscored value means that if the operand is omitted, the macro instruction will be expanded as though the underscored value had been coded. The underscored value is the default value

  ```
  ECB=INTERNAL | event_control_block_addr
  ```

In this example, INTERNAL is the default value. If the ECB operand is omitted, the assembler assumes ECB=INTERNAL.

Default values apply *only* to declarative macro instructions, and to the list (MF=L) or generate (MF=G) forms of TPL-based macro instructions. For all other macro instructions, and the modify (MF=M) or execute (MF=E) forms of TPL-based macro instructions, no default values are assumed. In the latter case, only the values specified in the macro instruction itself or already contained in the referenced control blocks are used.

- Brackets ([ ]) denote optional operands or values. Conversely, the lack of brackets indicates that an operand is required.

  In this example, the positional operand VERBATIM | SUMMARY is optional, and the MF operand is mandatory:

  ```
  [VERBATIM | SUMMARY,] MF=(E,tpl_address)
  ```

- An ellipsis (...) indicates that whatever precedes it (either an operand value or an entire operand) can be repeated any number of times.

  An operand appearing as

  ```
  (data structure name,...)
  ```

  could be coded as (TPL,TIB) if two data structure names are required, or as (TPL) or TPL if only one data structure name is required.

- Parentheses, equal signs, and commas must be coded exactly as shown, subject to the previously described rules.

Macro Instruction Operand Types

All macro instruction operands are either keyword or positional operands. Most of the API macro instruction operands are keyword operands.

Keyword Operands

Keyword operands consist of these items:

- A fixed character string (the operand keyword)
- An equal sign (=)
- A single or multiple operand value.

The presence of the equal sign distinguishes keywords from positional operands.

Keyword operands do not have to be coded in the order shown in the operands column of the format description.

Example

A macro instruction having these operands indicated in the operands column

```
[,DABUF=user_data_address]
[,DALEN=user_data_length]
```

could be coded as DABUF=BUFFER,DALEN=200 or as DALEN=200,DABUF=BUFFER

Keyword operands must be separated by commas. If a keyword operand is omitted, the commas that would have been included with it are also omitted.

There are a few instances in the API macro instructions where more than one value can be coded after the keyword. When this is done, operand sublist notation is used.

The option code operand specified as

```
OPTCD= ([SHORT | LONG]
         [,SYNC | ASYNC]
         [,NEGOT | NONEGOT]
         [,UNCOND | COND]
         [,DEBUG | NODEBUG])
```

could be coded as OPTCD=ASYNC or as OPTCD=(ASYNC) when only one option code is used.

When more than one option code is used, however, the option code names must be enclosed in parentheses and separated by commas, in this format:

```
OPTCD=(SHORT,ASYNC,NONEGOT,UNCOND,DEBUG)
```

If a field name is omitted, the comma that would have been included with it is also omitted.

Omitting the first, third, and fourth option code names in this example results in this format:

```
OPTCD=(ASYNC,DEBUG)
```

Positional Operands

Positional operands are used to a much lesser extent, and must be coded in the exact order shown in the operands column when more than one positional operand is shown.

Positional operands are separated by commas as are all operands, but if a positional operand is omitted, the surrounding commas must still be coded to indicate its absence.

A macro instruction that has the three positional operands OPTIONS, ADDRESS, and LENGTH would be coded as

```
OPTIONS,ADDRESS,LENGTH
```

if all three are coded, or as

```
,ADDRESS,LENGTH
```

if only the last two are coded.

If the last positional operand or operands are omitted, the trailing comma or commas should not be coded.

Operand Coding Order

Positional operands are generally coded before any keyword operands, but this is not required as long as the previous rules are followed. Also, all operands may be coded on a single line, separate lines, or a combination of both. When coding operands on more than one line, the standard assembler language rules for continuation lines apply.

List, Generate, Modify, and Execute Forms

The standard form (in other words, no MF operand indicated) of a nondeclarative API macro instruction expands into both nonexecutable code that represents the parameters specified on the macro instruction, and executable code that causes the API routines to be entered when the macro instruction is executed. The nonexecutable code, called the parameter list, is assembled at the point in the application program where the macro instruction appears. For TPL-based macro instructions, this parameter list is usually a short form TPL.

Standard Macro Instruction Disadvantages

Strict use of standard form macro instructions simplifies the design of an application program, but has these major disadvantages:

- Because the parameter list is expanded in-line with executable code, the application is rendered non-reentrant.
- Parameter lists for TPL-based macro instructions cannot be shared.

Alternative API Macro Instructions

Alternative forms of the API macro instructions are provided which overcome one or both of the disadvantages described in the preceding section. These various forms cause the assembler to respond in one of these ways:

- Build the parameter list where the macro instruction appears in the application program, but assemble no executable code (nonreentrant list form)
- Assemble code that builds the parameter list at a location indicated at execution time, but assembles no executable code to cause the API routines to be entered (reentrant list form)
- Assemble code that builds the parameter list at a location indicated at execution time, and assembles code that causes the appropriate API routine to be entered to execute the requested function (generate form)
- Assemble code that modifies an existing parameter list at execution time, but assembles no executable code to cause the API routines to be entered (modify form)
- Assemble code that modifies an existing parameter list and causes the appropriate API routine to be entered to execute the requested function (execute form)

The following tables summarize the various macro instruction forms.

Actions Taken for Various Macro Instruction Forms

This table lists the actions taken during assembly and execution for each form, and how it is specified on the macro instruction.

Table 1-1 Actions Taken for Various Macro Instruction Forms

	Actions Taken		
Form	**During Assembly**	**During Execution**	**Coded With**
Standard	Parameter list and code to execute function assembled where macro instruction appears in application program	In-line parameter list modified and requested function executed	no MF operand or MF=I

Table 1-1 Actions Taken for Various Macro Instruction Forms (Continued)

	Actions Taken		
Form	**During Assembly**	**During Execution**	**Coded With**
List (Non-reentrant)	Parameter list assembled where macro instruction appears in application program	No executable code (execute form must be used)	MF=L
List (Reentrant)	Executable code assembled to build parameter list at location indicated by application program	Parameter list built, but requested function not executed (execute form required)	MF=(L,address)
Generate	Code assembled to build parameter list and execute requested function	Parameter list built at location indicated by application program and requested function executed	MF=(G,address)
Modify	Code assembled to modify parameter list indicated by application program	Existing parameter list modified but requested function not executed (execute form required)	MF=(M,address)
Execute	Code assembled to modify parameter list and execute requested function	Existing parameter list modified and requested function executed	MF=(E,address)

Note The various alternative forms of the API macro instructions are designated with the MF operand.

Runtime Characteristics of Various Macro Instruction Forms

This table lists the runtime characteristics of each form, whether defaults apply to unspecified operands, and how operands are generated by the assembled code.

Table 1-2 Runtime Characteristics of Various Macro Instruction Forms

	Runtime Characteristics:			
Form	**Reentrant**	**Shared TPL**	**Defaults Apply**	**Operand Type**
Standard	no	no	yes	Address Constant
List (Non-reentrant)	no	yes	yes	Address Constant
List (Reentrant)	yes	yes	yes	Register Displacement
Generate	yes	no	yes	Register Displacement
Modify	yes	yes	no	Register Displacement
Execute	yes	yes	no	Register Displacement

The operand types the same as the ones used in the IBM Macro manual:

A - Address Constant Type
RX - Register Displacement Type

List Form

The MF operand for the list form of any non-declarative macro instruction is coded in this format:

MF = {L | (L address)}

L — Indicates that this is the list form of the macro instruction. If the format MF=L is coded, then the parameter list is assembled in place. If the format MF=(L,address) is coded, then the parameter list is built during program execution at the specified address. When the list form is used, default values are generated for any unspecified macro instruction operands.

address — Indicates the storage location where the parameter list is to be built during program execution. This area must begin on a fullword boundary and if the application program is to be reentrant, must be in dynamically allocated storage. Because the assembler builds executable code that in turn builds the parameter list, the macro instruction must be in the executable portion of the application program.

Generate Form

The MF operand for the generate form of any non-declarative macro instruction is coded in this format:

MF = (G,*address*)

G — Indicates that this is the generate form of the macro instruction. Default values are generated for any unspecified macro instruction operands.

address — Indicates the storage location of where the parameter list is to be built. Generally, this is in dynamically allocated storage. This area must begin on a fullword boundary and if the application program is to be reentrant, must be in dynamically allocated storage. Because the assembler builds executable code that in turn builds the parameter list, the macro instruction must be in the executable portion of the application program. After the parameter list is built, the requested function executes.

Modify Form

The MF operand for the modify form of any non-declarative macro instruction is coded in this format:

MF=(M,*address*)

M — Indicates that this is the modify form of the macro instruction. Default values are not generated for any unspecified macro instruction operands.

address — Indicates the storage location of an existing parameter list to be modified. Code is assembled to modify those parameters corresponding to operands specified on the macro instruction. If an operand is not specified, the parameter is not modified and no default value is assumed.

The modify form of a macro instruction lets the application program modify a parameter list after it has been built and before it is reused to execute another transport function. However, the parameter list can not be expanded. If the parameter list is a short form TPL and the macro instruction attempts to modify a parameter beyond the range of the parameter list, execution of the macro instruction is terminated with a general return code of 8 in register 15.

Execute Form

The MF operand for the execute form of any non-declarative macro instruction is coded in this format:

MF = (E,*address*)

E — Indicates that this is the execute form of the macro instruction. Default values are not generated for any unspecified macro instruction operands.

address — Indicates the storage location of an existing parameter list to be modified. Code is assembled to modify those parameters corresponding to operands specified on the macro instruction. If an operand is not specified, the parameter is not modified and no default value is assumed. After the parameter list is modified, the requested function executes.

For TPL-based macro instructions, the address indicated with the MF operand is the TPL itself. As an alternative to the execute form of the MF operand, TPL=address may be specified. In other words, TPL=address is equivalent to specifying MF=(E,address). This alternative is provided for VTAM programmers who may prefer the similarity to the RPL operand used in VTAM macro instructions. The TPL and MF operands must not be specified together.

Optional and Required Operands

The assembler format description shows most macro instruction operands as being optional. However, whether or not an operand is optional depends on the form of macro instruction used. Execute (MF=E) and modify (MF=M) forms update an existing parameter list, and therefore, all operands except the address of the parameter list itself are optional. The list forms (MF=L) assume that the parameter list may be updated later when it is executed with the execute form, and do not require all operands to be specified when the list is generated. Only the standard (MF=I) and generate (MF=G) forms require certain operands to be specified at assembly time, and even these are limited.

Since all API service functions (except TOPEN) require a valid endpoint identifier, the EP operand must be coded on the standard and generate forms of macro instructions.

The fact that an operand is optional for a given instance of a macro instruction does not mean the corresponding parameter is optional for the request. It is the responsibility of the application program to make sure that all required parameters have been included in the parameter list when it is executed. The API does extensive checking of the parameter list, and terminates processing of a request if any required parameters are missing.

Default and Maximum Values

The default values assumed by a macro instruction for unspecified operands only apply to the standard, list, and generate forms. In this case, the macro instruction is expanded as if all unspecified operands were coded with their default values. For modify and execute forms, only operands that are specified are modified. If an operand is not specified, the value stored in the parameter list is used during the subsequent execution of the requested function.

The method used to store the operand value varies depending on the form used. When an in-line parameter list is specified with the standard and nonreentrant list forms, A-type address constants are generally expanded which generate the operand values. After assembly, the operand values are contained at the proper locations within the parameter list.

For all other forms, and when register notation is not specified, a Load (L) or Load Address (LA) assembler instruction is usually expanded to load the value into register 14 or 15. The resulting value is stored at the proper location in the parameter list.

The use of a load address instruction has implications for the maximum value that can be specified using the latter forms. Address values are limited to the size of an address in the current addressing mode (24 or 31 bits), and quantity values are limited to 12 bits (4,095 decimal) unless an index or base register is specified, in which case it is limited to the size of an address. There is no restriction when register notation is used.

If LENGTH is an operand with an integer value, then the LENGTH=number argument is limited to a 12-bit value, the LENGTH=displacement(index,base) argument is limited to a 24-bit or 31-bit value, and the LENGTH=(register) argument can be a 32-bit value.

Rules for Loading Operand Values

Whether a Load (L) or Load Address (LA) instruction is used to load an operand value depends on the nature of the operand. These rules apply:

- If the operand is a simple integer value, or quantity (for example, the length or size of some object), it is loaded with a Load Address (LA) instruction.
- If the operand value is the address of some object, and the object itself can be generated at assembly time in static storage (for example, an ECB address, the address of an exit routine, or a storage area containing a protocol address), it is loaded with a Load Address (LA) instruction.
- All other operand values (for example, endpoint identifiers, TCB or ASCB addresses, and sequence numbers returned by the API) are loaded with a Load (L) instruction. These are generally values acquired from some other source.

Appendix B: Macro Instruction Operand Summary lists the operand formats for all API macro instructions, as well as the default values that apply when operand values are not specified.

Linkage Conventions

TPL-based macro instructions (standard, generate, and execute forms) all use the same conventions to call the API routines.

Register Contents on Routine Entry

On entry to the API routine, registers are set as shown in Runtime Characteristics of Various Macro Instruction Forms.

Table 1-3 Register Contents on Routine Entry

R0	Function code.
R1	Address of the parameter list (TPL).
R2-R12	Unmodified application program registers.
R13	Address of 18-word register save area.

Table 1-3 Register Contents on Routine Entry (Continued)

R14	Address of the next sequential instruction in the application program.
R15	The API entry point address.

Register Contents on Return

On return from the API routine, the registers are set as shown in Register Contents on Return.

Table 1-4 Register Contents on Return

R0	Recovery action or conditional completion code.
R1	Address of parameter list (TPL).
R2 - R12	Unmodified application program registers.
R13	Address of 18-word register save area.
R14	Address of the next sequential instruction in the application program.
R15	General return code (0 if successful).

The function code passed in register 0 may be negated to indicate some function-specific option. Also, the values returned in register 0 and 15 may be modified by the SYNAD or LERAD exit routine if the request completed abnormally. Appendix C: Register Usage Summary summarizes register usage conventions employed by the API.

Description of Macro Instructions

This section lists detailed descriptions of all API macro instructions, arranged in alphabetical order. Each macro instruction description includes this information:

- The name of the macro instruction
- A brief statement of its function and use
- The assembler format description
- A detailed description of each operand
- A description of completion information returned
- A table of return codes
- General usage information

It is assumed that you are familiar with the API concepts and facilities presented in *Cisco IOS for S/390 Assembler API Concepts*.

ACLOSE

Terminate Session with the API Subsystem

The ACLOSE macro instruction is used to terminate a session between the application program and the API. The APCB used to establish the session is the sole operand of the ACLOSE macro instruction.

Syntax Description

[*symbol*] ACLOSE *APCB_address*

APCB_address Indicates the address of an opened APCB (Application Program Control Block) that defines the session with the API. Unless supplied in a general register, the address of the APCB is loaded into register 1 using a Load Address (LA) instruction. If register notation is specified, the address of the APCB may be contained in any one of the general registers 1-12. Only one APCB can be closed with the ACLOSE macro instruction. An invalid or corrupted value causes unpredictable results. Default: none (must be specified)

Completion Information

After the ACLOSE macro instruction completes successfully, the APCB is closed and the session established when the APCB was opened is terminated. Fields modified by the API when the APCB was opened are returned to their original values, and any resources allocated by the API on behalf of the transport user are released. In particular, any transport endpoints associated with the APCB are closed. If ACLOSE completes unsuccessfully, the session is not terminated.

When control is returned to the next sequential instruction following the ACLOSE macro instruction, successful completion is indicated by a return code of 0 in register 15, and an error code of 0 in register 0. The error code field in the APCB is also set to 0. If the ACLOSE macro instruction is unsuccessful, an error code is returned in register 0, and in most cases, is also stored in the APCB. Unsuccessful completion is indicated by one of these non-zero return codes.

Table 1-5 describes the ACLOSE Unsuccessful Completion Return Codes.

Table 1-5 ACLOSE Unsuccessful Completion Return Codes

4 (X'04')	The APCB was already closed. The error code in register 0 is set to APCBECLS, and the APCB is unmodified.
12 (X'0C')	The ACLOSE macro instruction failed and the error is permanent. The error code returned in register 0 is also stored in the APCB, and indicates the reason for the failure. The permanent error flag is not set in the APCB.
16 (X'10')	A fatal error occurred and the APCB could not be closed. An error code is returned in register 0, but the error code field in the APCB is not changed. This return code is generally the result of an invalid APCB.

The value returned in register 0 indicates the nature of the error encountered by the ACLOSE macro instruction.

Return Codes

If the APCB was determined to be valid, was not already closed and was not busy, the API also returns this value in the APCBERRC field of the APCB. An APCB is considered busy if it is in the process of being opened or closed by another task. This table lists these error codes, which are defined in more detail in the *Cisco IOS for S/390 Unprefixed Messages and Codes* manual.

Table 1-6 APCB Error Codes for ACLOSE

Return Code	(Register 15)	Specific Error Code (Register 0)		
0	(X'00')	0		
4	(X'04')	APCBECLS		
12	(X'0C')	APCBEPRB APCBETRV	APCBELER APCBEENV	APCBEAMD APCBEEND
16	(X'10')	APCBELER	APCBEVCK	APCBEBSY

Usage Information

The ACLOSE macro instruction closes (or "deactivates") an APCB and terminates the session that was established when the APCB was opened. Any endpoints opened by the transport user are closed, and all resources associated with the APCB are released. No more endpoints can be opened by the transport user. Fields within the APCB that were filled in by the API when the APCB was opened are reset to their original value, and the APCB should not be referenced unless reopened by another AOPEN macro instruction.

The ACLOSE macro instruction must be issued in the mainline program. That is, the ACLOSE macro instruction must be executed from a PRB, and no IRBs or SVRBs may exist in the current RB chain. The addressing mode in effect must also be consistent with the APCB when it was opened. If the APCB was opened in 31-bit mode, and RMODE=ANY was coded on the APCB (the default), the APCB must also be closed in 31-bit mode. There is only one form of the ACLOSE macro instruction, and its expansion is always reentrant. If the task that opened an APCB terminates before closing the APCB, an ACLOSE is issued by the API task termination exit.

AOPEN

Establish Session with the API Subsystem

The AOPEN macro instruction establishes a session between the application program and the API. Parameters that describe the application program, and specify the access method to be used, are provided in an APCB. The address of the APCB is the sole operand of the AOPEN macro instruction.

[*symbol*] AOPEN *APCB_address*

Syntax Description

APCB_address Indicates the APCB to be associated with the transport user issuing the AOPEN macro instruction. Unless supplied in a general register, the address of the APCB is loaded into register 1 using a Load Address (LA) instruction. If register notation is specified, the address of the APCB may be contained in any one of the general registers 1-12. Only one APCB can be opened with the AOPEN macro instruction. Default: none (must be specified)

Completion Information

After the AOPEN macro instruction completes successfully, the APCB is initialized and a session is established with the API. Certain fields within the APCB are modified during AOPEN processing to reference data areas used by the API. The application program should not modify any information within the APCB while it is opened. These fields are restored to their original values when the APCB is closed.

When control is returned to the next sequential instruction following the AOPEN macro instruction, successful completion is indicated by a return code of 0 in Register 15, and an error code of 0 in Register 0. The error code field in the APCB is also set to 0.

If the AOPEN macro instruction completes unsuccessfully, an error code is returned in register 0, and in most cases, is also stored in the APCB. Unsuccessful completion is indicated by one of these non-zero return codes.

Table 1-7 describes the AOPEN Unsuccessful Completion Return Codes.

Table 1-7

4 (X'04')	The APCB was already opened. The error code in register 0 is set to APCBEOPN, and the APCB is not modified.
8 (X'08')	The AOPEN macro instruction failed because of some temporary condition. The error code returned in register 0 is also stored in the APCB, and indicates the reason for the failure. The AOPEN macro instruction can be retried at some later time.
12 (X'0C')	The AOPEN macro instruction failed and the error is permanent. The error code returned in register 0 is stored in the APCB and indicates the reason for the failure. The permanent error flag is also set in the APCB, and must be cleared before the APCB can be used with another AOPEN macro instruction.
16 (X'10')	A fatal error occurred and the APCB could not be opened. An error code is returned in register 0, but the error code field in the APCB is unchanged. This return code is generally the result of an invalid APCB.

The value returned in register 0 indicates the nature of the error encountered by the AOPEN macro instruction. If the APCB was determined to be valid, was not already opened, and was not busy or flagged with a permanent error, the API also returns this value in the APCBERRC field of the APCB.

This table lists these error codes; they are defined in more detail in the *Cisco IOS for S/390 Unprefixed Messages and Codes* manual.

Table 1-8 APCB Error Codes for AOPEN

Return Code	(Register 15)	Specific Error Code (Register 0)		
0	(X'00')	0		
4	(X'04')	APCBEOPN		
8	(X'08')	APCBERDY	APCBECVT	
12	(X'0C')	APCBEPRB APCBECFG APCBESTP APCBEOPT	APCBELER APCBEVCK APCBEDRA APCBEENV	APCBEACT APCBEBEG APCBEVER APCBEDUP
16	(X'10')	APCBELER APCBEBSY	APCBEVCK	APCBEPER

Usage Information

The AOPEN macro instruction opens (or "activates") an APCB, and establishes a session between the application program and the API. The APCB defines a specific transport user (in other words, the issuing task), and is used to associate subsequent service requests with this transport user. The APCB also serves as an anchor for the API data structures and transfer vectors that are required to execute subsequent requests.

When the AOPEN macro instruction completes successfully, fields within the APCB contain information stored by the API. This information is used to process future service requests and should not be modified by the application program while the APCB is open. When the APCB is closed, these fields are returned to their original, pre-opened condition. Modification of any of these fields while the APCB is open causes unpredictable results.

The API runs as a separate job in its own address space. Because an application program can be started before the API, an application program may issue an AOPEN macro instruction before the API is active. In this case, the AOPEN fails, and the application program is informed that the API is inactive. These possibilities exist, each of which is indicated by a separate error code:

- The API subsystem has not been configured in the MVS operating system, either because it has not yet been installed, or it has not been activated since the last IPL.
- The API subsystem is configured, but is not active. This is considered a permanent error, and the application programmer should check with the system operator to determine why the API is not active.
- The API subsystem is active, but has not completed initialization, and is not ready to service requests. This is considered a temporary error, the AOPEN macro instruction can be retried after some delay.
- The API subsystem is ready, but the access method interface has not completed initialization. This error is similar to the previous error, and should be retried after some delay.

  The addressing mode in effect at the time the AOPEN macro instruction is executed determines the residency mode of data areas allocated by the API. The location of the APCB must also be consistent with the current addressing mode.

- If the addressing mode is 24-bit, all dynamic storage is allocated with LOC=BELOW, and the APCB must reside below 16MB. Future service requests can then be executed in any addressing mode.
- If the addressing mode is 31-bit, all storage is allocated with LOC=ANY, and all future service requests must be issued in 31-bit addressing mode. The APCB can reside above or below 16MB.

Note The RMODE parameter on the APCB can be used to force allocation below 16MB, thus allowing the APCB to be opened in 31-bit mode, and future requests to be issued in 24-bit mode.

The AOPEN macro instruction must be issued in the mainline program (in other words, the AOPEN macro instruction must be executed from a PRB, and no IRBs or SVRBs may exist in the current RB chain). There is no limit on the number of operand APCBs for an address. There is no limit on the number of tasks per address space that can have currently-opened APCBs. Each APCB defines a different transport user.

There is only one form of the AOPEN macro instruction, and its expansion is always reentrant. However, since AOPEN modifies the APCB, the APCB may need to be moved to dynamically-allocated storage before AOPEN is executed if the application program is to be reentrant.

APCB

Generate an Application Program Control Block

The APCB macro instruction is used to generate an Application Program Control Block (APCB). The APCB identifies a transport user, and contains information that is used to service requests issued by the transport user. The address of the APCB is supplied as an operand of an AOPEN and ACLOSE macro instruction. The address of an opened APCB must also be provided whenever a new transport endpoint is created.

[*symbol*] APCB [AM = TLI]
[,EXLST = *exit_list_address*]
[,APPLID = *application_name*]
[,PASSWD =*application_password*]
[,SYSID = *Cisco IOS for S/390_API_subsys_name*]
[,ENVIRO = ASM | IBMC | SASC]
[,ACNTX = *application_level context*]
[,ECNTX =*environment_level_context*]
[,OPTCD = ([TRACE | NOTRACE]
[,AUTHEXIT | NOAUTHEXIT])]
[,RMODE=24 | ANY]
[,MF = L | DSECT]

Syntax Description

AM = TLI

Indicates the access method that is used by the application program. TLI must be coded as shown when using the transport layer interface.

Default: TLI (Transport Layer Interface)

EXLST = *exit_list_address*

Specifies the address of an exit list to be associated with the APCB. The exit list contains addresses of user exit routines to be entered when certain events occur. The exit list can be generated by the TEXLST macro instruction, and should specify AOPEN, indicating that the exit list is to be linked to the transport user via AOPEN.

An exit list identified by the APCB applies to all endpoints opened by the transport user. A separate exit list can be provided for each endpoint by identifying the exit list with TOPEN. The same exit list can be referenced by more than one APCB. If no exit list is specified, the application program can not receive asynchronous notification when the corresponding events occur. In this case, the application program must process such events synchronously by analyzing the return code at the completion of each service function.

Default: 0 (no APCB exit list)

APPLID = *application_name*

Identifies the name of the application program. The name must be coded as an alphanumeric string up to eight characters in length. If the name is longer than 8 characters, it is truncated to eight bytes. The application name is used in combination with the password to authorize access to the API services. If a user ID is not provided when endpoints are opened, the application name is also used to authorize endpoint services.

If no name is provided, a null string of eight zero bytes is generated. If this field is still set to zero when the APCB is opened, the API substitutes the step name from the TIOT. This is either the procedure step name if the job step was invoked via a procedure, or the job step name otherwise.

Default: null name (use TIOT step name)

PASSWD = *application_password*

Identifies the password associated with the application name. The password may be any alphanumeric string up to eight characters in length. If the password is longer than eight characters, it is truncated to eight bytes. If the password operand is not coded, a null password is generated consisting of eight zero bytes.

Default: null password (no password provided)

SYSID =
Cisco IOS for S/390_API_subsys_name

The Cisco IOS for S/390 API runs as an MVS subsystem, and is initially located via its subsystem name when the APCB is opened with an AOPEN macro instruction. This subsystem name is specified when the API is installed. If the subsystem name defined during installation does not agree with the default name used by the APCB macro instruction, or if more than one instance of the API can run on the local system, the subsystem name can be specified by this operand.

The subsystem name is an alphanumeric string up to four characters in length. If the subsystem name specified is longer than four characters, it is truncated to four bytes. If no subsystem name is coded, the default name is used.

Default: ACSS

ENVIRO = ASM | IBMC | SASC

Identifies the runtime environment for the application program. Generally, the runtime environment is assembler language, and ASM should be coded. IBMC and SASC are reserved for the API library routines which execute in the runtime environment of the IBM and SAS/C compilers, respectively. This operand selects special interface exits that initialize and terminate the runtime environment, and that schedule user exit routines written in a higher-level language.

Normal application programs should either leave this operand unspecified, or always specify ENVIRO=ASM. Specifying other environments yields unpredictable, and unsatisfactory, results.

Default: ASM (assembler language environment)

ACNTX = *application_level_context*

Specifies an arbitrary fullword of user context associated with the application program. This information is not interpreted by the API, and is merely saved in the APCB, and is included in the parameter list provided to exit routines. This information can be used by the application program to derive application-level context associated with the transport user during exit processing. Any value that can be generated as an A-type address constant can be specified.

Note The value stored in the APCB is moved to another data area after the APCB is opened. If the application program changes the value in the APCB after AOPEN has been executed, this is not reflected in the value passed to the exit routine.

Default: 0 (no application-level context specified)

ECNTX = *environment_level_context*

Specifies an arbitrary fullword of user context associated with the runtime environment of the application program. This information is not interpreted by the API, and is merely saved in the APCB, and is included in the parameter list provided to exit routines. Any value that can be generated as an A-type address constant can be specified.

Note The value stored in the APCB is moved to another data area after the APCB is opened. If the application program changes the value in the APCB after AOPEN has been executed, this is not reflected in the value passed to the exit routine.

This information is intended to be used by built-in interface routines that map the assembler language runtime environment of the API into the runtime environment of the application program.

If ENVIRO=ASM is coded, this field can be used for any purpose by the application program.

If ENVIRO=ASM is not coded, this field is reserved for use by the run-time environment exits.

Default: 0 (no environment-level context specified)

OPTCD = TRACE | NOTRACE

Indicates whether or not service requests issued by the transport user associated with this APCB are to be traced by the API.

If the option is OPTCD=TRACE, events associated with endpoints linked to this APCB are to be traced.

If the option is OPTCD=NOTRACE, tracing is disabled for this transport user.

If OPTCD=TRACE is selected when the APCB is opened, a storage area is allocated within the application program's address space for tracing endpoint events. All events associated with a given transport user are co-mingled in time-order sequence. These events are traced:

- Transport service function invoked
- Service request rejected
- WAIT SVC issued by TCHECK
- Service request completed
- Asynchronous completion exit entered
- Asynchronous event exit entered
- Synchronous error exit entered
- TCHECK control function completed
- TERROR control function completed
- TSTATE control function completed

The occurrence of an event is recorded by an eight-word trace entry stored in a circular trace table. The information saved in each trace entry depends on the type of event. Exit events save the first six words of the TXP associated with the exit. All other events save the first five words of the TPL associated with the event, and its address. All events contain a time-stamp generated by a store clock (STCK) instruction.

Default: TRACE (trace endpoint events)

OPTCD = AUTHEXIT | NOAUTHEXIT

Indicates if exits associated with endpoints or with the API session are to be driven in fast-path, or authorized, mode.

If the option is OPTCD=AUTHEXIT, exits are given control in SRB mode, key 0, and supervisor state.

If the option is OPTCD=NOAUTHEXIT, exits are driven in IRB or PRB mode, caller key, and problem state.

Default: NOAUTHEXIT

RMODE = 24 | ANY

Indicates the residency mode of any storage allocated in the application program's address space by the API.

If the option is RMODE=24, storage is always allocated below 16MB. Local data areas used by the API are always allocated below 16MB and are always addressable no matter what addressing mode is in effect.

If the option is RMODE=ANY, storage is allocation in accordance with the current addressing mode. Storage is allocated above 16MB if the APCB is opened in 31-bit mode, and below 16MB otherwise. Therefore, the API service functions associated with an APCB opened in 31-bit mode and RMODE=ANY must be executed in 31-bit mode.

The addressing mode of all subsequent service requests associated with this APCB must be consistent with the addressing mode in effect when the APCB is opened, as well as the RMODE parameter of the APCB itself.

The application program is also responsible for assuring that the residency mode of data areas it manages (for example, the APCB and TPLs) is compatible with the addressing mode. The API performs consistency checks on the addressing mode whenever a service request is issued. However, it is possible that unpredictable results can occur before the API has had an opportunity to perform this check.

Default: ANY (allocate storage above 16M)

MF = L | DSECT

Indicates the form of the macro expansion.

If the option is MF=L, The APCB is generated in-line with the APCB macro instruction.

If the option is MF=DSECT, a dsect that maps the fields of the APCB is generated.

There is no remote list form of the APCB macro instruction. If the application program is reentrant and must generate an APCB in dynamic storage, it should allocate the storage area and move a copy of the APCB into it. The skeleton APCB can be generated in static storage as long as it is not opened with the AOPEN macro instruction.

Default: MF=L (in-line list)

Completion Information

The APCB macro instruction is declarative and generates no executable instructions. Refer to the description of the AOPEN and ACLOSE macro instructions for completion information.

Return Codes

The APCB contains fields in which return codes are stored during AOPEN and ACLOSE processing. If either of these macro instructions complete with an error, error information generally is returned in these fields. APCBERRC contains a one-byte specific error code, and APCBDGNC contains a two-byte diagnostic code. The codes that can be returned in these fields are defined in the *Cisco IOS for S/390 Unprefixed Messages and Codes* manual.

Usage Information

Each application program must define a transport user before it can obtain the services of a transport provider. A transport user is defined by creating an APCB containing information required by the API and then activating the APCB by referencing it in an AOPEN macro instruction. The task that issues the AOPEN macro instruction becomes permanently associated with the APCB and is considered the transport user.

Opening the APCB causes the API to create the necessary infrastructure for servicing the transport user, and pointers to various components of the infrastructure are stored in the APCB. This information must not be modified while the APCB is opened. The APCB serves as an anchor for all information associated with the transport user and the address of this APCB must be provided every time the transport user opens a new endpoint.

An application program can have more than one transport user, and each transport user must be associated with a unique APCB. If two instances of the API happen to be running on the local system (each with a different subsystem name), and an APCB is opened by the same task for both, the transport user defined by one APCB is completely independent of the other.

The APCB can also be thought of as defining a session between the application program and the API. The AOPEN macro instruction establishes the session, and the ACLOSE macro instruction terminates the session. Once a session is established, the transport user can request services such as opening endpoints, binding protocol addresses, and transferring data. When the APCB is closed, the session is terminated and such requests can no longer be made. Any endpoints opened by the transport user are closed and all resources allocated to the transport user are released.

The APCB serves as an anchor for the API data areas and contains information about the transport user. An exit list of routines to be entered when certain events occur is also linked to the APCB. An arbitrary word of user context stored in the APCB is provided with each entry to any of these exit routines. This information is not examined or interpreted by the API.

The APCB can reside in 24-bit or 31-bit storage. The application program is required to assure that the location of the APCB and any data areas it references is consistent with current addressing mode. In particular, neither the executable instructions expanded by API macro instructions nor the interface routines they call, change the current addressing mode, and operate in the addressing mode of the caller. It is generally advisable that all future requests made to the API be issued in the same addressing mode in effect when the APCB is opened.

TACCEPT

Accept a Connection Request

When a connect indication has been received at an endpoint with a TLISTEN macro instruction, the TACCEPT macro instruction is used to accept the connection request, and to establish a connection with the remote transport user. The connection can be established to a new endpoint, or to the endpoint on which the TLISTEN was executed.

[*symbol*] TACCEPT [EP = *endpoint_id*]
[,NEWEP = *new_endpoint_id*]
[,SEQNO =*sequence_number*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E,[*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TACCEPT macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

NEWEP = *new_endpoint_id*

Indicates the endpoint at which the accepted connection is established. The operand value is the endpoint identifier returned by a TOPEN macro. A zero value indicates the listening endpoint (in other words, the endpoint at which the connect indication arrived).

The application program can accept a connection on the listening endpoint or a newly created endpoint. If accepted on the listening endpoint, the endpoint must not have any pending connect indications other than the one being accepted. The endpoint must be in the connect-indication-pending state (TSINCONN), and have only one indication pending in its queue. Otherwise, the TACCEPT macro instruction is completed abnormally.

If the connection is accepted on a new endpoint, the endpoint must be in the disabled state (TSDSABLD), and must have been opened by the task that opened the listening endpoint. The local protocol address bound to the endpoint may be the same address bound to the listening endpoint, or different. Connecting to an endpoint with a different protocol address may not be supported by all transport providers.

Default: 0 (connection established to listening endpoint)

SEQNO = *sequence_number*

Specifies which connect indication is being accepted. The value specified must have been returned by a TLISTEN macro instruction. The transport provider uses this value to identify a connect indication pending for this endpoint which has not yet been accepted or rejected.

Default: 0 (most likely an invalid sequence number)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TACCEPT macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TACCEPT request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TACCEPT macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TACCEPT macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TACCEPT macro instruction. The second sublist operand, *tpl_address*, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TACCEPT macro instruction completes normally (or conditionally) when the accepted connection has been established.

- If the connection is accepted to itself, the state of the endpoint is changed from connect-indication-pending (TSINCONN) to connected (TSCONNCT), and the endpoint is ready to send and receive data.
- If the connection is accepted to a new endpoint, the state of the new endpoint is changed from disabled (TSDSABLD) to connected (TSCONNCT), and the old endpoint can continue receiving connect indications.
- If only one connect indication was pending, the state of the old endpoint is changed from connect-indication-pending (TSINCONN) to enabled (TSENABLD). Otherwise, the state of the endpoint is unchanged.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY), and a conditional completion code is returned in register 0. The TPL return code field is set accordingly. No other information is returned.

If the TACCEPT macro instruction completes abnormally, no connection is established, and the connect indication remains queued. If the connection request was abandoned via a disconnect, the indication is removed from the queue when the TCLEAR macro instruction is executed. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TACCEPT return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-9 TACCEPT Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY		
TRFAILED	TAEXCPTN	TENONEGO		
	TAINTEG	TEPROTO	TEDISCON	
	TAENVIRO	TESYSERR TESTOP TEUNSUPF	TESUBSYS TETERM TEUNAUTH	TEDRAIN TEUNSUPO TERSOURC
	TAFORMAT	TEBDFNCD TEBDEXIT TEBDSQNO	TEBDOPCD TEBDDATA TEBDEPID	TEBDECB TEBDOPTN
	TAPROCED	TEAMODE TENOCONN TEOWNER	TESTATE TEINDICA	TEINCMPL TEACCEPT
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TACCEPT macro accepts a pending connect indication that has been received at an endpoint. A sequence number returned by a previous TLISTEN macro instruction identifies which connect indication is being accepted in the event more than one is pending. On the successful completion of this request, the specified endpoint is connected, and is ready to send and receive data.

The connection can be established to the endpoint at which the connect indication was received, or established to a new endpoint. If established to the endpoint that issued the corresponding TLISTEN macro instruction, no more than one connect indication may be pending. The maximum number of

pending connect indications is limited to the queue length specified when the endpoint was enabled (see TBIND). Once the connection has been established, the listening endpoint becomes busy, and does not generate any additional connect indications. However, connection requests arriving at the endpoint may continue to be queued by the transport provider.

Application programs that establish a connection to the listening endpoint operate as single-threaded servers. Multi-threaded servers, on the other hand, must leave the endpoint available to receive additional connect indications. In this case, a new endpoint must be created for each connection that is established. The endpoint must exist in the same communications domain, and employ the same type of service as the listening endpoint. The local protocol address bound to the new endpoint is generally the same as the listening endpoint.

The endpoint identifier provided with the TACCEPT macro instruction identifies the endpoint to which the connection is established. If the value specified for NEWEP is zero, the connection is established to the listening endpoint (if possible). If the endpoint ID identifies a different endpoint, the endpoint must have been opened by the same task that opened the listening endpoint, and must be in the disabled (TSDSABLD) state.

The TACCEPT macro instruction is normally used to establish connections to endpoints operating in connection mode. However, if the endpoint is operating in connectionless mode, and was enabled to simulate connect indications, the TACCEPT macro instruction can be used to create an association with a transport user whose protocol address was returned with the TLISTEN macro instruction. Read *Cisco IOS for S/390 Assembler API Concepts* manual for a discussion on associations in connectionless mode.

TADDR

Retrieve Local or Remote Protocol Address

The local protocol address bound to an endpoint, or the remote protocol address of a connected (or associated) transport user, can be retrieved using the TADDR macro instruction.

[*symbol*] TADDR [EP = *endpoint_id*]
[,ADLEN = *protocol_address_length*]
[,ADBUF = *protocol_address_address*]
[,ADALET = *protocol_address_alet*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,TRUNC | NOTRUNC]
[,LOCAL | REMOTE])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TADDR macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

ADLEN = *protocol_address_length*

Indicates the length (in bytes) of the protocol address storage area identified by the ADBUF operand. The length is updated when the request is completed to reflect the actual length of the protocol address returned. If the length is zero, the TADDR macro instruction is abnormally completed.

Default: 0 (return no protocol address)

ADBUF = *protocol_address_address*

Indicates the address of a storage area for returning the protocol address of the designated transport user. The storage area should be large enough to contain the entire address. The format of the protocol address is provider-dependent, and its maximum size can be determined by issuing a TINFO macro instruction. The storage area can be aligned on any boundary.

Default: 0 (no protocol address storage area)

ADALET = *protocol_address_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the ADBUF parameter. The ADALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The ADALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TADDR macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TADDR request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = TRUNC | NOTRUNC

Indicates whether the protocol address returned to the application program by the transport provider should be truncated if it does not fit within the storage area provided.

If the option is OPTCD=TRUNC, the excess is truncated, and the TADDR macro instruction is completed conditionally as long as no other errors occur.

If the option is OPTCD=NOTRUNC, nothing is placed in the storage area, and the TADDR macro instruction is completed abnormally.

Default: NOTRUNC (no truncation)

OPTCD = LOCAL | REMOTE

Indicates which protocol address to return to the application program.

If the option is OPTCD=LOCAL, the protocol address of the local transport user, which was bound to the endpoint by the application program, is returned.

If the option is OPTCD=REMOTE, the protocol address of the remote transport user connected to, or associated with, the endpoint is returned.

Default: LOCAL (local protocol address)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TADDR macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB. The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TADDR macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TADDR macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TADDR macro instruction completes normally (or conditionally) when the designated protocol address has been returned to the application program. The length of the storage area is updated to reflect the actual length of the protocol address. The negotiated length of the connect indication queue is also returned in the QLSTN field of the TPL associated with this request. The state of the endpoint is not changed.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY), and a conditional completion code is returned in register 0. TCTRUNC is set if the protocol address returned to the application program was truncated to fit in the storage area provided. The TPL return code field is set accordingly. No other information is returned.

If the TADDR macro instruction completes abnormally, no information is returned. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TADDR return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-10 TADDR Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY	TCTRUNC	
TRFAILED	TAINTEG	TEOVRFLO		
	TAENVIRO	TESYSERR TESTOP	TESUBSYS TETERM	TEDRAIN
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD TEBDADDR	TEBDECB
	TAPROCED	TEAMODE	TESTATE	TEINCMPL
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TADDR macro instruction is used to retrieve the protocol address of the local or remote transport user associated with an endpoint. The local protocol address can be retrieved any time after a TBIND macro instruction has been successfully executed at the endpoint. The remote protocol address can be retrieved any time after a TACCEPT or TCONFIRM macro instruction has been successfully executed, completing the connection or association with a remote transport user.

Generally, this information is returned to the application program by the macro instruction that bound the local protocol address, or completed the connection or association. However, if the application program did not provide a storage area in which the protocol address could be returned, or needs to acquire the information again, the TADDR macro instruction can be used. An example of the latter is after an endpoint, which has already been bound to a local protocol address, is passed to another task or address space. The negotiated length of the connect indication queue can also be acquired in this manner.

If the TADDR macro instruction is executed at an endpoint operating in connectionless mode, either the bound local protocol address, or the remote protocol address associated with the last received datagram, is returned as indicated by the OPTCD operand.

When a local protocol address is retrieved, a partial protocol address may be returned (in other words, a portion of the protocol address may be unknown at the time the TADDR macro instruction was issued, and the corresponding field is set to zero). In particular, if a TADDR macro instruction is issued before a COTS endpoint is connected, the network address portion of the protocol address may be set to zero. This is because multi-homed hosts (in other words, hosts with more than one network connection) cannot determine the local network address until the destination is known. In the case of a CLTS endpoint, the local network address may change if datagrams are transmitted or received via different network connections.

TBIND

Bind a Protocol Address to a Transport Endpoint

The TBIND macro instruction is used to bind a local protocol address to an endpoint, and to define the number of connections that can be pending for the endpoint. When completed, a COTS endpoint is ready to begin connection establishment, and a CLTS endpoint is ready to begin data transfer.

[*symbol*] TBIND [EP=*endpoint_id*]
[,ADLEN = *protocol_address_length*]
[,ADBUF = *protocol_address_address*]
[,ADALET = *protocol_address_alet*]
[,QLSTN = *listen_queue_length*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,TRUNC | NOTRUNC]
[,NEGOT | NONEGOT]
[,ASSIGN | USE])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [tpl_address])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TBIND macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

ADLEN = *protocol_address_length*

Indicates the length (in bytes) of the protocol address, or protocol address storage area, identified by the ADBUF operand. If the storage area is for retrieving a protocol address from the transport provider, the length indicated is the maximum length of the storage area. Otherwise, this operand indicates the length of the protocol address contained in the storage area. The length may be zero, indicating there is no protocol address or storage area.

Default: 0 (no protocol address)

ADBUF = *protocol_address_address*

Indicates the address of a protocol address storage area that contains either a local protocol address assigned by the application program, or a local protocol address assigned and returned by the transport provider. Option codes indicated with the OPTCD operand determine how the storage area is used. If the local protocol address is to be assigned by the transport provider, the storage area must be large enough to contain the returned address. The format and maximum size of a protocol address is provider-dependent, and can be determined by issuing a TINFO macro. The storage area can be aligned on any boundary.

Default: 0 (no protocol address storage area)

ADALET = *protocol_address_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the ADBUF parameter. The ADALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The ADALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

QLSTN = *listen_queue_length*

Indicates the size of the queue for holding incoming connections arriving at the endpoint, and pending connect indications received by the application program.

If the value specified is zero, no connections can be queued, and the endpoint is disabled for receiving connects.

If the value specified is non-zero, incoming connections are queued, and corresponding connect indications are generated at the endpoint.

A connect indication remains pending until it is accepted or rejected by the application program, or until the connection is abandoned by the caller. The value of this operand generally determines whether the application program is operating in client or server mode.

The transport provider may not be able to queue the number of connections specified by the application program, and as a result, attempts to negotiate the indicated value to a lesser amount. If negotiation is permitted by the application program (OPTCD=NEGOT), the request completes conditionally, and returns a conditional completion code in register 0. Otherwise, the TBIND request completes abnormally.

Default: 0 (endpoint is disabled)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TBIND macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TBIND request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = TRUNC | NOTRUNC

Indicates whether the protocol address returned to the application program by the transport provider should be truncated if it does not fit within the storage area provided.

If the option is OPTCD=TRUNC, the excess is truncated, and the TBIND macro instruction is completed conditionally as long as no other errors occur.

If the option is OPTCD=NOTRUNC, nothing is placed in the storage area, and the TBIND macro instruction is completed abnormally.

Default: NOTRUNC (no truncation)

OPTCD = NEGOT | NONEGOT

Indicates whether the value specified for QLSTN can be negotiated to a lesser value.

If the option is OPTCD=NEGOT (and the value is larger than can be accommodated by the transport provider), a smaller value is used. The negotiated value is returned to the application program, and the TBIND function completes conditionally as long as no other errors occur.

If the option is OPTCD=NONEGOT, no negotiation is performed, and the TBIND function completes abnormally.

Default: NONEGOT (negotiation disallowed)

OPTCD = ASSIGN | USE

Indicates whether the application program or the transport provider is to assign the local protocol address.

If the option is OPTCD=ASSIGN, the transport provider assigns an appropriate address. If ADLEN and ADBUF designate a storage area, the transport provider returns the protocol address assigned.

If the option is OPTCD=USE, the transport provider uses the protocol address provided by the application program. The storage area designated by the ADLEN and ADBUF operands must contain a valid protocol address.

Default: USE (use protocol address provided by application program)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TBIND macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TBIND macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TBIND macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TBIND function completes normally (or conditionally) when the local protocol address has been bound to the specified endpoint. The state of the endpoint is changed to disabled (TSDSABLD) or enabled (TSENABLD) in accordance with the value specified for QLSTN. Any protocol address assigned by the transport provider is returned in the storage area provided by the application program. The length of the storage area is updated to reflect the actual length of the assigned protocol address.

- If the endpoint is operating in connectionless mode, then it is ready to send and receive datagrams.
- If the endpoint is operating in connection mode and disabled, the endpoint can be used to initiate a connection request.
- If the endpoint is operating in connection mode and is enabled, the endpoint can receive connect indications generated by arriving connection requests.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY), and a conditional completion code is returned in register 0. TCNEGOT is set in the conditional completion code if the value specified for QLSTN was negotiated to a lesser value. The negotiated value is returned in the same TPL storage location. TCTRUNC is set if the storage area was too small to contain the entire protocol address returned by the transport provider. The TPL return code field is set accordingly. No other information is returned.

If the TBIND function completes abnormally, the state of the endpoint is unchanged. If the state of the endpoint was opened (TSOPENED), no protocol address is bound to the endpoint. If the state was disabled (TSDSABLD), the endpoint remains in its disabled condition, and the length of the connect indication queue is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TBIND return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-11 TBIND Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY	TCTRUNC	TCNEGOT
TRFAILED	TAEXCPTN	TENONEGO		
	TAENVIRO	TESYSERR TESTOP TERSOURC	TESUBSYS TETERM TEINUSE	TEDRAIN TEUNAUTH
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD TEBDADDR	TEBDECB TEBDQLEN
	TAPROCED	TEAMODE	TESTATE	TEINCMPL
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		

Table 1-11 TBIND Return Codes (Continued)

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.

Usage Information

The TBIND macro instruction associates a local protocol address with a transport endpoint. Once a protocol address has been bound, an application program may begin sending and receiving datagrams through the endpoint if operating in connectionless mode, or may request a connection to a remote transport user if operating in connection mode. Optionally, a connection-mode endpoint can be enabled for receiving connection requests from a remote transport user.

The local protocol address may be assigned by the application program or the transport provider.

- If assigned by the application program (OPTCD=USE), then the protocol address is placed in a storage area designated by the ADLEN and ADBUF operands, and is passed to the transport provider for binding to the endpoint.
- If assigned by the transport provider (OPTCD=ASSIGN), then the storage area is used to return the assigned address.

Generally, an application operating in server mode assigns the protocol address, and an application operating in client mode defers this responsibility to the transport provider.

The application program can only assign the local transport address; the local network address is always assigned by the transport provider. Therefore, when assigning a protocol address, the application program should set the network address field to zero, or provide only a partial (in other words, truncated) protocol address. Similarly, the protocol address returned by the API contains a null network address. The local network address is not assigned until a connection has been established, or a datagram has been received at the endpoint. If the application program requires the local network address, it should execute a TADDR macro instruction after the connection has been established at a COTS endpoint, or after a datagram has been transmitted or received at a CLTS endpoint.

Note The local network address bound to a connectionless-mode endpoint may change with each datagram transmitted or received via the endpoint.

Application programs operating in client mode are active, and as soon as a local protocol address is bound, the endpoint can be used to initiate a connection to the server. Application programs operating in server mode are passive, and wait for the client to send a connection request. Arriving connections are queued by the transport provider, and cause connect indications to be generated at the endpoint. The number of connections that can be queued at the endpoint, and the number of connect indications that can be pending, is determined by the value of QLSTN specified when the local protocol address was bound.

Connections that arrive at the endpoint are queued until accepted or rejected by the application program. When the queue is full, subsequent connection requests are discarded. A larger value for QLSTN reduces the chances of a transport user finding the endpoint busy, and gives the application more time to respond to a connect indication. A value of zero disables the endpoint from receiving any connect indications. Application programs operating in client mode must leave the endpoint disabled by setting QLSTN to zero.

Note The value of QLSTN does not limit the number of transport users that can be simultaneously connected to the application program. The number of pending connections is limited by the QLSTN value; the number of established connections is limited by the number of requests the application program is willing to accept. One endpoint must be created for each accepted connection (see TACCEPT).

The value specified for QLSTN by application programs operating in connectionless mode should normally be zero. However, if a non-zero value is specified, the transport provider simulates connection-mode service (in other words, the first arriving datagram generates a connect indication, which the application program is expected to receive via a TLISTEN macro instruction, and accept or reject with a TACCEPT or TREJECT macro instruction). If accepted, an association is created for the endpoint, and TRECV and TSEND macro instructions may be used to send and receive data. If the server is multi-threaded, datagrams arriving from other transport users generates new connect indications.

The binding of the local protocol address and enabling of the endpoint may be separated into two distinct requests issued to the transport provider. If the first request binds a protocol address and leaves the endpoint disabled by specifying a QLSTN value of zero, a second TBIND macro instruction may be executed to enable the endpoint. If the second instance of TBIND specifies a protocol address, it must be the address already bound to the endpoint.

The format and content of a protocol address is provider-dependent. To minimize dependence on a particular provider, or particular protocol, applications operating in server mode should use directory services to map service names into local protocol addresses. Clients should let the transport provider assign the protocol address, and should use directory services to obtain addresses of remote hosts and services. Chapter 2: DNR Directory Services provides information on using Network Directory Services (NDS) supported by Cisco IOS for S/390.

TCHECK

Check Status of a Transport Service Request

The TCHECK macro instruction is used to check the completion status of an active TPL. If necessary, a system WAIT macro instruction is issued to wait for completion, and if the request completed abnormally, the SYNAD or LERAD exit routine is entered.

[*symbol*] TCHECK MF = (E, *tpl_address*)

Syntax Description

MF = (E, *tpl_address*)

Indicates the execute form of the TCHECK macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL associated with the request whose completion status is being checked.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: none (must be coded as indicated)

Completion Information

The TCHECK macro instruction completes normally or conditionally when the request associated with the TPL has been posted complete, and the TPL has been set inactive. The state of the endpoint is updated to reflect the actions taken by the API routines. If an internal or external ECB is being used for synchronization, the ECB is cleared.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY) to indicate successful completion, and the conditional completion code (if any) is returned in register 0. The TPL return code field is set accordingly. Other information may be returned in the TPL or storage areas provided by the application program in accordance with the particular service requested by the TPL.

If the TCHECK macro instruction completes abnormally, the TPL associated with the request is set inactive, the internal or external ECB (if any) is cleared, and the state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

The return codes passed in registers 0 and 15, and stored in the TPL return code field, are set in accordance with the particular request associated with the TPL. The return codes that are valid for each macro instruction are listed with the description of the macro instruction. The return codes listed in this table are those that can be generated during TCHECK processing:

Note If the SYNAD or LERAD exit routine is scheduled, the values in register 0 and 15 at the next sequential instruction following the macro instruction are the values returned by the exit routine.

This table lists the symbolic names for the TCHECK return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-12 TCHECK Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation
TRFAILED	TAPROCED	TEINACTV
	TATPLERR	TEB4EXIT
TRFATLFC	func. code	The function code loaded into register 0 was invalid.
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.

Usage Information

The TCHECK macro instruction is used to check the completion status of a TPL-based request executed in asynchronous mode. If the request is incomplete, the task issuing the TCHECK macro instruction is suspended until posted complete. The TPL is set inactive so it can be used with another request, and if the request completed abnormally, the SYNAD or LERAD exit routine (if any) is entered.

When a TPL-based request is executed in asynchronous mode (in other words, OPTCD=ASYNC was indicated by macro instruction), the API returns to the application program immediately after scheduling the requested function. If the return code in register 15 is 0 (TROKAY), the request was accepted, and is executed asynchronously. The application program must subsequently issue a TCHECK macro instruction to check for completion of the request.

Request completion is indicated in one of these ways:

- Posting of an ECB
- Entering a TPL exit routine

The mechanism used for a particular request is indicated by the ECB or EXIT macro instruction operands, and the actions performed by the TCHECK macro instruction depend on which was indicated.

When the TCHECK macro instruction is executed for a TPL-based request that specified an ECB, these actions occur:

- If the request being checked has not been completed, execution of the application program task that issued the TCHECK macro instruction is suspended until the request has been completed. However, any exit routines associated with the endpoint can still run.
- The ECB (internal or external) is cleared.

Note The ECB specified in the TPL-based macro instruction must not be cleared between the time the macro instruction is issued and the corresponding TCHECK macro instruction is issued. If the ECB is cleared during this interval, the application program task may be suspended indefinitely.

- The TPL being checked is marked inactive and available for reuse by another TPL-based macro instruction.
- If the request being checked has been completed normally or conditionally (as indicated by the return code), control is returned to the next sequential instruction following the TCHECK macro instruction.
- If the request being checked has been completed abnormally, the SYNAD or LERAD exit routine is invoked, if available; otherwise, control is returned to the next sequential instruction following the TCHECK macro instruction.
- If the SYNAD or LERAD exit routine is invoked, and the exit routine returns to the address in register 14 at the time it was entered, control is returned to the next sequential instruction following the TCHECK macro instruction. In this case, the contents of register 0 and 15 are those returned by the exit routine.

When a TPL exit routine is being used in place of an ECB, the TCHECK macro instruction should be issued within the exit routine. These actions occur:

- The TPL being checked is marked inactive and available for reuse by another TPL-based macro instruction.
- If the request being checked has been completed normally or conditionally (as indicated by the return code), control is returned to the next sequential instruction following the TCHECK macro instruction.
- If the request being checked has been completed abnormally, the SYNAD or LERAD exit routine is invoked, if available; otherwise, control is returned to the next sequential instruction following the TCHECK macro instruction.
- If the SYNAD or LERAD exit routine is invoked, and the exit routine returns to the address in register 14 at the time it was entered, control is returned to the next sequential instruction following the TCHECK macro instruction. In this case, the contents of register 0 and 15 are those returned by the exit routine.
- If issued outside the exit routine, and before the exit routine is invoked, the TCHECK macro instruction completes abnormally.

A TCHECK request can only be issued against a TPL that is marked active. Abnormal completion occurs if issued for a TPL that is inactive. The TCHECK macro instruction should never be issued for a TPL that is associated with a synchronous request. When operating in synchronous mode, the API performs TCHECK equivalent processing automatically, and processing proceeds as though an internal ECB had been specified.

TCLEAR

Acknowledge (Clear) Disconnect Indication

The TCLEAR macro instruction is issued to clear a disconnect indication pending at an endpoint, and to receive the disconnect reason code and any data accompanying the disconnect.

[*symbol*] TCLEAR [EP = *endpoint_id*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TCLEAR macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TCLEAR macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TCLEAR request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TCLEAR macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TCLEAR macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TCLEAR macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TCLEAR macro instruction completes normally (or conditionally) when the pending disconnect indication has been cleared, and any disconnect user data received with the disconnect has been moved into the storage area provided by the application program. The length of the storage area is updated to reflect the actual amount of disconnect user data returned.

The state of the endpoint is changed in accordance with the value specified for QLSTN by a TBIND macro instruction. If the endpoint cannot receive connect indications, the state is changed to disabled (TSDSABLD). Otherwise, the state is changed to enabled (TSENABLD) unless other connect indications are pending, in which case the state is unchanged (in other words, the current state must have been connect-indication-pending (TSINCONN).

A protocol-dependent reason code is returned in the DISCD field of the TPL which specifies the reason for the disconnect. If the disconnect was received for a pending connect indication that is being abandoned by the remote transport user, the associated sequence number is returned in the TPLSEQNO field of the TPL.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY), and a conditional completion code is returned in register 0. TCTRUNC is set if the disconnect user data returned to the application program was truncated to fit in the storage area provided. The TPL return code field is set accordingly. No other information is returned.

If the TCLEAR macro instruction completes abnormally, no information is returned, and the disconnect indication (if any) remains queued. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TCLEAR return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-13 TCLEAR Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY	TCTRUNC	
	TAINTEG	TEPROTO	TEOVRFLO	
	TAENVIRO	TESYSERR TESTOP TEUNSUPF	TESUBSYS TETERM	TEDRAIN TEUNSUPO
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD TEBDDATA	TEBDECB
	TAPROCED	TEAMODE TENODISC	TESTATE	TEINCMPL
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TCLEAR macro instruction is used to clear a pending disconnect indication, and to receive any user data sent with the disconnect request. This macro instruction should only be issued if a disconnect indication exists at the endpoint.

A disconnect indication is generated when a disconnect request is received at an endpoint, and can occur under any one of the following conditions:

- A connection request initiated by a TCONNECT macro instruction is pending, and the called transport user rejected the request.
- A connection request initiated by a TCONNECT macro instruction was accepted by the called transport user, but before the application issued a TCONFIRM macro instruction completing the connection, the transport user disconnected.

- A connect indication that was received with a TLISTEN macro instruction is pending, and the calling transport user disconnected before it was accepted with a TACCEPT macro instruction, or rejected with a TREJECT macro instruction.
- A connection was fully established, and the peer transport user disconnected while data transfer was in progress.
- Data transfer was complete, but before the application program could disconnect, the peer transport user requests a disconnect.
- The application program initiated an orderly release with the TRELEASE macro instruction, and the peer transport user disconnected before the complementary orderly release was received with the TRELACK macro instruction.
- The peer transport user initiated an orderly release that was received by a TRELACK macro instruction, but disconnected before the application program completed the orderly release with a TRELEASE macro instruction.

These disconnect indications are all initiated as the result of some action by the remote transport user. The transport provider may also generate a disconnect indication on its own. This generally occurs when the transport provider fails to establish or maintain a connection because of a protocol error or network malfunction.

A disconnect causes immediate release of a connection. If the connection was fully established, any data buffered at the endpoint is discarded, and no additional data can be sent. The only information queued at the endpoint is the disconnect user data that was sent with the disconnect request. Once a disconnect indication is pending, only a TCLEAR (or TCLOSE) macro instruction should be issued.

A limited amount of user data may be received with the disconnect indication if supported by the transport provider. The content of this data is application-dependent, and not interpreted by the API or the transport provider. The maximum length of user data that can be received can be determined by issuing a TINFO macro instruction. Disconnects initiated by the transport provider are never accompanied by user data.

A disconnect reason code is returned by the transport provider. The value and interpretation of the reason code is provider-dependent. These are some possible reasons:

- Invoked by remote transport user (disconnect user data may contain additional information)
- Invoked by the transport provider due to the lack of resources
- Quality of service below minimum level
- Invoked by the transport provider due to protocol error or fatal malfunction
- Called transport user is unknown or unreachable
- Called transport user is unavailable
- Invoked by transport provider, but no reason given

If a disconnect occurs on an enabled endpoint while a connect indication is pending, a disconnect indication is generated. The indication can be presented synchronously with the completion of a TACCEPT (or TREJECT) macro instruction, or generated asynchronously by scheduling the DISCONN exit (see TEXLST). In either case, when the TCLEAR macro instruction is executed, a sequence number is returned. The sequence number should be used by the application program to determine which connection request to abandon.

The TCLEAR macro instruction is normally only executed at endpoints operating in connection mode. However, if an association has been established for an endpoint operating in connectionless mode, the transport provider may generate a disconnect indication under certain abnormal conditions. In this case, the indication must be cleared with a TCLEAR macro instruction, which also serves to terminate the association.

TCLOSE

Close a Transport Endpoint

The TCLOSE macro instruction is used to close an endpoint, and alternatively, to pass control of the endpoint to another task or address space. If the endpoint is to be deleted, any resources associated with the endpoint are released.

[*symbol*] TCLOSE [EP = *endpoint_id*]
[,TCB = *task_control_block_address*]
[,ASCB = *address_space_control_block_address* | ANY]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,DELETE | PASS])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TCLOSE macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

TCB = *task_control_block_address*

Indicates the TCB address of a task that acquires control of the endpoint when OPTCD=PASS is specified.

If the indicated value is zero, control is passed to the first task in the specified address space, which issues a complementary TOPEN macro instruction.

If the indicated value is not zero, control can only be passed to the indicated task.

Default: 0 (pass to any task)

ASCB = *address_space_control_block_address* | ANY

Indicates the ASCB address of another address space that acquires control of the endpoint when OPTCD=PASS is specified.

If the indicated value is zero, the endpoint can only be passed to a task executing in the same address space.

If the indicated value is not zero, the indicated value is the ASCB address of another address space containing the task that acquires control.

If the indicated value is ANY, the endpoint can be passed to any address space in the system.

Default: 0 (pass to task in this address space)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TCLOSE macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TCLOSE request. The application program is responsible for issuing the TCHECK macro instruction to complete the TCLOSE request.

Default: SYNC (synchronous mode)

OPTCD = DELETE | PASS

Indicates the disposition of the endpoint designated in the TPL associated with this macro instruction.

If the option is OPTCD=DELETE, the endpoint is closed, and any record of the endpoint is deleted from all internal tables and local storage.

If the option is OPTCD=PASS, the endpoint is not closed, and control of the endpoint is passed to the designated task or address space that issued the corresponding TOPEN (OPTCD=OLD).

When control is being passed, the TCLOSE request does not complete until a complementary TOPEN (OPTCD=OLD) macro instruction has been issued by the acquiring task or address space.

Default: DELETE (delete endpoint)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TCLOSE macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TCLOSE macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TCLOSE macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TCLOSE macro instruction completes normally when the endpoint has been closed, or control has been passed to the acquiring task.

- If the OPTCD=DELETE operand was indicated, the state of the endpoint is changed to closed (TSCLOSED).
- If the OPTCD=PASS operand was indicated, the state of the endpoint is unchanged.

In this case, the TCLOSE macro instruction does not complete until the acquiring task successfully executes a matching TOPEN macro instruction indicating OPTCD=OLD, at which time it becomes the controlling task.

The TCB and ASCB addresses of the acquiring task are returned in the TPL.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY). The conditional completion code in register 0 is always 0 (TCOKAY), and the TPL return code field is set accordingly. No other information is returned.

If the TCLOSE macro instruction completes abnormally, the issuing task continues to control the endpoint. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TCLOSE return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-14 TCLOSE Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY		
TRFAILED	TAENVIRO	TESYSERR TESTOP	TESUBSYS TETERM TERSOURC	TEDRAIN TEUNAUTH TENOTACT
	TAFORMAT	TEBDOPCD TEBDTCB	TEBDXECB TEBDASCB	TEBDEPID
	TAPROCED	TEAMODE	TEOWNER	TEINCMPL TESTATE
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TCLOSE macro instruction is used to terminate (in other words, close) an endpoint, and to remove any record of it from the system and communications domain in which it was created. Alternatively, the TCLOSE macro instruction may be used to pass control to an endpoint within the same task, or to another task or address space.

If OPTCD=DELETE is indicated, the TCLOSE macro instruction closes the endpoint, and removes any record of it from the system. All internal resources allocated to the endpoint are deleted, and the endpoint can no longer be referenced by any TPL-based macro instruction. Its existence in the communications domain, specified when the endpoint was created, is terminated.

An endpoint can be closed from any state. Normally, the application program closes an endpoint after any connection or association established at the endpoint has been released, and the local protocol address has been unbound. In this case, the state of the endpoint is opened (TSOPENED). However, if the endpoint is closed from any other state, the previously described operations are simulated to return the endpoint to the opened state before closing. If the endpoint was engaged in data transfer, any data buffered at the endpoint is discarded, and an abortive disconnect is executed. It is also possible to issue TCLOSE OPTCD=DELETE on the endpoint that is being passed by TCLOSE OPTCD=PASS.

The TCLOSE macro instruction provides a quick and convenient way to clean up after a major failure. In fact, the API resource recovery routines executes a TCLOSE function for each endpoint opened by a terminating task. The application program is responsible for recovering resources that it has allocated such as TPL and data storage areas.

An endpoint can only be closed by the task that opened it. The opening task is said to control (or own) the endpoint, although other tasks may issue the API macro instructions that reference the endpoint. If it is necessary for another task to close an endpoint, control must be passed by indicating OPTCD=PASS when executing the TCLOSE macro instruction. The task acquiring control must issue a complementary TOPEN macro instruction indicating OPTCD=OLD, which specifies the same endpoint. The TCB and ASCB addresses of the new controlling task are returned to the application program.

Note The endpoint identifier may change as control is passed from one task to another (see TOPEN). If control of an endpoint is passed to another task in the same address space, the task relinquishing control may continue to reference the endpoint, but must use the new endpoint identifier. The new endpoint ID is created and returned to the acquiring task after the complementary TOPEN macro instruction completes. The old endpoint ID should never be referenced again, either by the relinquishing or acquiring task. Please note that there are no restrictions of the order in which TCLOSE OPTCD=DELETE and TOPEN OPTCD=OLD occur.

Control can be passed to any task or address space that would otherwise have been authorized to create the endpoint.

- If a TCB address is indicated with the TCLOSE macro instruction, then only that task can acquire control.
- If an ASCB address is indicated, then only a task in the designated address space can acquire control.
- If neither are indicated, then the first task in the current address space to issue the complementary TOPEN macro instruction acquires control of the endpoint.

The endpoint can be passed in the opened (TSOPENED), disabled (TSDSABLD), enabled (TSENABLD), or connected (TSCONNCT) state, and must not have any incomplete requests pending at the endpoint. The state of the endpoint is unchanged, and the task relinquishing control may continue to reference the endpoint using the new endpoint ID created by the corresponding TOPEN OPTCD=OLD. Any exit list associated with the endpoint is replaced by the new exit list (if any) indicated by the complementary TOPEN macro instruction.

Whenever an asynchronous TCLOSE (OPTCD=ASYNC) completes normally (in other words, recovery code is TAOKAY), the TCLOSE must be TCHECKED to let the cleanup of endpoint related control blocks complete. TOPEN specified with OPTCD=OLD does not complete until a TCHECK is done on the related asynchronous TCLOSE OPTCD=PASS.

TCONFIRM

Acknowledge Confirm Indication

A connect confirm indication received at an endpoint as the result of a previous TCONNECT request is acknowledged with the TCONFIRM macro instruction. The remote protocol address of the connected (or associated) transport user is returned to the application program.

[*symbol*] TCONFIRM [EP = *endpoint_id*]
[,ADLEN = *protocol_address_length*]
[,ADBUF = *protocol_address_address*]
[,ADALET = *protocol_address_alet*
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,TRUNC | NOTRUNC]
[,BLOCK | NOBLOCK])]
([ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E,[*tpl_address*]),

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TCONFIRM macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

ADLEN = *protocol_address_length*

Indicates the length (in bytes) of the protocol address storage area identified by the ADBUF operand. The length is updated when the request is completed to reflect the actual length of the protocol address returned. If the length is zero, the protocol address of the called transport user is not returned to the application program.

Default: 0 (return no protocol address)

ADBUF = *protocol_address_address*

Indicates the address of a storage area for returning the protocol address of the called transport user. The storage area should be large enough to contain the entire address. The format of the protocol address is provider-dependent, and its maximum size can be determined by issuing a TINFO macro instruction. The storage area can be aligned on any boundary.

Default: 0 (no protocol address storage area)

ADALET = *protocol_address_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the ADBUF parameter. The ADALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The ADALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TCONFIRM macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TCONFIRM request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = TRUNC | NOTRUNC

Indicates whether the information returned to the application program by the transport provider should be truncated if it does not fit within the storage area provided.

If the option is OPTCD=TRUNC, the excess is truncated, and the TCONFIRM macro instruction is completed conditionally as long as no other errors occur.

If the option is OPTCD=NOTRUNC, nothing is placed in the storage area, and the TCONFIRM macro instruction is completed abnormally.

Default: NOTRUNC (no truncation)

OPTCD = BLOCK | NOBLOCK

Indicates whether or not the issuing task can be suspended if the TCONFIRM macro instruction cannot be completed immediately.

If the option is OPTCD=BLOCK (and no connect information has been received), the issuing task is suspended until the connection request is confirmed.

If the option is OPTCD=NOBLOCK, the TCONFIRM macro instruction is completed immediately, and an abnormal return code indicates that the task would have been suspended for an indefinite period of time.

In either case, if a connect confirmation has already been received, the TCONFIRM macro instruction completes normally without suspending the issuing task.

When OPTCD=NOBLOCK is indicated, the TCONFIRM macro instruction can be used to poll for a connect confirmation. If the connection request has already been confirmed, the request is completed as usual. Otherwise, the request is completed abnormally, and the transport user can try again after delaying an appropriate period of time (for example, the expected round-trip time.

Default: BLOCK (suspend issuing task if necessary)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TCONFIRM macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TCONFIRM macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TCONFIRM macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TCONFIRM macro instruction completes normally (or conditionally) when the confirm indication, generated by a connect confirmation sent by the called transport user, has been received by the application program. The state of the endpoint is changed from connect-in-progress (TSOUCONN) to connected (TSCONNCT), and the endpoint is ready to send and receive data.

If a storage area was provided by the application program, the protocol address of the called transport user is returned. The protocol address length field indicated by macro instruction operand ADLEN is updated to reflect the actual length of the protocol address.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY), and a conditional completion code is returned in register 0. TCTRUNC is set if the protocol address returned to the application program was truncated to fit in the storage area provided. The TPL return code field is set accordingly. No other information is returned.

If the TCONFIRM macro instruction completes abnormally, no information is returned, and the pending connect confirmation (if any) remains queued. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TCONFIRM return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-15 TCONFIRM Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY	TCTRUNC	
TRFAILED	TAEXCPTN	TENOBLOK		
	TAINTEG	TEPROTO	TEOVRFLO	TEDISCON
	TAENVIRO	TESYSERR TESTOP TEUNSUPF	TESUBSYS TETERM	TEDRAIN TEUNSUPO
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD TEBDDATA	TEBDECB TEBDOPTN
	TAPROCED	TEAMODE	TESTATE	TEINCMPL
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		

Table 1-15 TCONFIRM Return Codes (Continued)

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.

Usage Information

The TCONFIRM macro instruction is used to receive a confirm indication that is pending for the endpoint. A confirm indication is generated when a connect confirmation is sent by a remote transport user in response to a connection request initiated by the application program with a TCONNECT macro instruction. On successful completion of this macro instruction, the specified endpoint is connected, and ready to send and receive data.

The protocol address of the remote transport user is returned to the application program.

Note Existing protocols require that the called protocol address (the protocol address supplied with the TCONNECT request) and responding protocol address (the protocol address returned with TCONFIRM completion) be the same.

The TCONFIRM macro instruction is normally issued at an endpoint operating in connection mode. However, if the endpoint is operating in connectionless mode, and the application program created an association by issuing a TCONNECT macro instruction, the TCONFIRM macro instruction must be issued to receive the (simulated) connect confirmation generated by the API. Refer to *Cisco IOS for S/390 Assembler API Concepts* for a discussion of associations in connectionless mode.

TCONNECT

Request a Connection with another Transport User

The TCONNECT macro is used to request a connection to a remote transport user. The application program supplies the remote protocol address, protocol options, and any connect user data it wants to send to the remote transport user.

[*symbol*] TCONNECT [EP = *endpoint_id*]
[,ADLEN = *protocol_address_length*]
[,ADBUF = *protocol_address_address*]
[,ADALET = *protocol_address_alet*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | ME, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TCONNECT macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

ADLEN = *protocol_address_length*

Indicates the length (in bytes) of the protocol address contained in the storage area identified by the ADBUF operand. A length of zero is invalid, and causes the request to be abnormally completed.

Default: 0 (no protocol address)

ADBUF = *protocol_address_address*

Indicates the address of a storage area containing the protocol address of a remote transport user.

If the endpoint is operating in connection mode, a connection is established to the transport user at the indicated protocol address.

If the endpoint is operating in connectionless mode, an association is made such that any datagram sent with a TSEND macro instruction is sent to the indicated address, and a TRECV macro instruction acts as a filter selecting only those datagrams from the associated protocol address.

The length of the protocol address is designated by the ADLEN operand.

Default: 0 (no protocol address storage area)

ADALET = *protocol_address_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the ADBUF parameter. The ADALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The ADALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TCONNECT macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TCONNECT request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TCONNECT macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TCONNECT macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TCONNECT macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TCONNECT macro instruction completes normally (or conditionally) when the connect-request primitive has been issued to the transport provider. The state of the endpoint is changed from disabled (TSDSABLD) to connect-in-progress (TSOUCONN). The connection is not established until a confirm indication has been received by the application program via a TCONFIRM macro instruction.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY), and a conditional completion code is returned in register 0. TCNEGOT is set in the conditional completion code if a requested protocol option was negotiated to an inferior value. The TPL return code field is set accordingly. No other information is returned.

If the TCONNECT macro instruction completes abnormally, no connection request is sent to the called transport user. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TCONNECT return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-16 TCONNECT Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY	TCNEGOT	
TRFAILED	TAEXCPTN	TENONEGO		
	TAINTEG	TEPROTO		
	TAENVIRO	TESYSERR TETERM TERSOURC	TESUBSYS TESTOP TEUNSUPF	TEDRAIN TEUNSUPO TEUNAUTH
	TAFORMAT	TEBDFNCD TEBDEXIT TEBDOPTN	TEBDOPCD TEBDADDR	TEBDECB TEBDDATA
	TAPROCED	TEAMODE	TESTATE	TEINCMPL
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TCONNECT macro instruction is used to request a connection with a remote transport user. The protocol address of the called transport user, connect user data, and protocol options to be associated with the connection are provided with the connection request. The endpoint must be in the disabled state (TSDSABLD) when the TCONNECT macro instruction is issued.

The TCONNECT macro instruction only serves to initiate the connection request. It is completed when the corresponding connect-request primitive has been issued to the transport provider. The connection is not established until a connect confirmation has been received from the called transport user. Receipt of a connect confirmation causes a confirm indication to be generated, and the application program can receive the indication with a TCONFIRM macro instruction.

The TCONNECT macro instruction is generally used by application programs operating in client mode. The protocol address (or name) of the remote server should be well known to the application program and other transport users that desire to use its services. Local directory services can be used to map host and service names into a protocol address that can then be supplied with the TCONNECT macro instruction. Directory services supported by Cisco IOS for S/390 are documented in Chapter 2, DNR Directory Services.

The TCONNECT macro instruction is primarily intended for endpoints operating in connection mode. However, when executed at an endpoint operating in connectionless mode, a permanent association is made with the indicated transport user. Once this association has been made, TSEND may be used in place of TSENDTO to transmit datagrams to the associated transport user. Also, TRECV may be used in place of TRECVFR to select datagrams received from the same transport user.

An association is most useful when the application program is operating as a client using a connectionless-mode service. The TCONNECT macro instruction is used in the same way as with connection-mode service. The protocol address and options are saved and used for each subsequent send and receive operation. A TCONFIRM macro instruction is required to complete the association, just as in connection mode.

TDISCONN

Initiate Abortive Disconnect

The TDISCONN macro instruction is used to release a connection to a remote transport user, or to abandon a connection attempt that is in progress. The connection release is immediate, and no attempt is made to preserve data in transit.

[*symbol*] TDISCONN [EP = *endpoint_id*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,ABORT | CLEAR])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TDISCONN macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or if MF operand is omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TDISCONN macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TDISCONN request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = ABORT | CLEAR

Indicates the action to be taken by the transport provider when the application program issues a TDISCONN macro instruction at an endpoint for which a disconnect indication is already pending.

If the option is OPTCD=ABORT, the TDISCONN request clears the disconnect indication, and the macro instruction is completed normally (or conditionally) if no other errors occur.

If the option is OPTCD=CLEAR, the request is completed abnormally, and the application program must clear the pending disconnect indication with a TCLEAR macro instruction.

Default: CLEAR (clear pending disconnect indication)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TDISCONN macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TDISCONN macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TDISCONN macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TDISCONN macro instruction completes normally when the corresponding disconnect primitive has been issued to the transport provider. Connection release is immediate.

The state of the endpoint is changed to disabled (TSDSABLD) or enabled (TSENABLD) in accordance with the value of QLSTN specified in the TBIND macro instruction.

- If the new state is Enabled, the endpoint can resume receiving connect indication.
- If the new state is Disabled, the endpoint can be used to request another connection.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY). The conditional completion code in register 0 is always 0 (TCOKAY), and the TPL return code field is set accordingly. No other information is returned.

If the TDISCONN macro instruction completes abnormally, the connection is not released or abandoned. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TDISCONN return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-17 TDISCONN Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY		
	TAINTEG	TEPROTO	TEDISCON	
	TAENVIRO	TESYSERR TESTOP TEUNSUPF	TESUBSYS TETERM	TEDRAIN TEUNSUPO
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD TEBDDATA	TEBDECB
	TAPROCED	TEAMODE	TESTATE	TEINCMPL
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TDISCONN macro instruction is used to request the immediate release of an established connection, or to abandon a pending connection request previously initiated with a TCONNECT macro instruction.

The TDISCONN macro instruction causes the abortive release of a connection. That is, the release is immediate, and any unsent data buffered at the endpoint is discarded. If it is important that all previously sent data reach the remote transport user, the application program should use an orderly release initiated with a TRELEASE macro instruction, or implement a session-layer protocol to gracefully terminate the session.

The TDISCONN macro instruction is normally issued when the endpoint is in the connected (TSCONNCT) state. The connection may have been established in client mode using the TCONNECT and TCONFIRM macro instructions, or established in server mode using the TLISTEN and TACCEPT macro instructions. When control is returned to the application program, the endpoint is considered disconnected, and is returned to the state that existed before the connection was established.

- If the application program is operating in client mode, the endpoint returns to the disabled (TSDSABLD) state, and another connection request can be initiated.
- If the application program is operating in as a multi-threaded server, the endpoint is also returned to the disabled state, and can be reused for accepting new connections.
- If the application program is operating as a single-threaded server, the endpoint is returned to the enabled (TSENABLD) state, and listening for another connect indication can resume.

The TDISCONN macro instruction may also be used to abandon a connection request that has not yet been confirmed. This situation may occur when the application program has issued a TCONNECT macro instruction, and has not received a connect confirmation.

Example

The remote transport user may be slow in responding, or the confirmation may be delayed because of congestion or other network problems. In this case, the endpoint must be in the connect-in-progress (TSOUCONN) state when the TDISCONN macro instruction is issued, and enters the disabled (TSDSABLD) state when the macro instruction completes.

Connected endpoints normally operate in connection mode. However, if an association has been established with a remote transport user, the endpoint may be operating in connectionless mode, and the TDISCONN macro instruction may be issued to terminate this association. The application program may then create a new association, begin sending and receiving datagrams with the TSENDTO and TRECVFR macro instructions, or unbind the protocol address and close the endpoint.

TDSECT

Generate the API Dummy Control Sections

The TDSECT macro instruction is used to generate dsects for the API data structures that must be created, managed, or referenced by the application program.

```
[symbol] TDSECT ([,TEM]
                 [,TIB]
                 [,TPA]
                 [,TPL]
                 [,TPO]
                 [,TPL]
                 [,TPO]
                 [,TSW]
                 [,TUB]
                 [,TPO]
                 [,TXL]
                 [,TXP]
                  [,ALL])
```

Syntax Description

dsect types

Indicates a list of dsects (dummy control sections) that should be generated during assembly of the application program. Each operand is the name of an API data structure.

This list describes the valid operands:

TEM - Defines the structure and content of an Transport Error Message returned by the TERROR macro instruction. The information returned is formatted as a multi-line WTO parameter list, and can be supplied directly to a WTO macro instruction. The application program may use this dsect to manipulate certain fields within the parameter list (for example, route codes and message descriptors).

TIB - Defines the structure and content of a Transport Protocol Information Block (TIB). The TIB contains basic transport protocol information returned by the TINFO macro instruction when OPTCD=PRIMARY is indicated. The format of this information is standard for all transport providers, and is intended to convey the basic characteristics of the underlying transport protocol and service.

TPA - Defines the structure and content of a Transport Protocol Address (TPA) in a particular communications domain. The domain can be specified with the DOMAIN operand, and should be consistent with domain specified when endpoints are opened (see TOPEN).

TPL - Defines the structure and content of a Transport Service Parameter List (TPL). The TPL is the primary API data structure for passing information between the application program and the API routines. All TPL-based macro instruction operands are stored or anchored in the TPL.

dsect types (continued)

TPO - Defines the structure and content of Transport Protocol Options (TPO) supported by a particular transport provider. The transport provider is identified by the communications domain that it services, and is indicated by the DOMAIN operand on the TDSECT macro instruction. The value specified should be consistent with the communications domain specified when endpoints are opened (see TOPEN).

TSW - Defines the structure and content of a Transport Endpoint State Word (TSW). The TSW contains endpoint state information that is returned by the TSTATE macro instruction.

TUB - Defines the structure and content of a Transport Endpoint User Block (TUB). The TUB contains user ID, group name, and password information that are associated with an endpoint for the purpose of authorizing access to facilities, and accounting for their use The TUB may be provided as an argument of the TOPEN and TUSER macro instructions.

TXL - Defines the structure and content of a Transport Endpoint Exit List (TXL). The TXL may be provided as a parameter of an AOPEN or TOPEN macro instruction, and is used to identify exit routines that are to be entered for processing certain asynchronous events.

TXP - Defines the structure and content of a Transport Exit Parameter List (TXP). The TXP contains parameters and other information passed by the API to exit routines that are entered to process asynchronous events. The address of the TXP is loaded into register 1 when the exit routine is entered.

ALL - This is not the name of a structure; it is an indication to generate all of the previous dsects as if each name had been indicated separately.

Note These names should not appear in the operand list more than once, and should not be included on more than one TDSECT macro instruction.

Completion Information

The TDSECT macro instruction is declarative and does not generate any executable code. The indicated dsects are generated at the point in the application program where the TDSECT macro instruction occurs.

Return Codes

No return codes are generated.

Usage Information

The TDSECT macro instruction is used to generate dummy control sections (dsects) that map the API data structures. Each operand of the macro instruction is the name of an API data structure, and is also the label that should appear in a USING statement to establish addressability to the data structure.

The API defines several data structures that are shared between the API routines and the application program. Some of these structures are created by the application program and referenced by the API, and others are created by the API and referenced by the application program. In all cases, the API routines use the same dsects to access and manipulate these data structures. If the application program does likewise, information that is bound at assembly time can be easily changed by reassembly of the application program.

If the application program manipulates fields in the TPL directly instead of using the API macro instructions, the TPL dsect should always be used to reference storage locations. Also, whenever one of these data structures is created dynamically, the symbolic name for the length of the data structure should be used for allocating or reserving memory. The API always follows the convention that the symbolic name that specifies the length is the structure name appended with the characters LEN.

Example

The length of a standard format TPL is defined by the symbol TPLLEN.

The application program should be careful not to use any assembly language labels that conflict with labels defined by these dsects. The API labels always begin with the letter T.

TERROR

Analyze Error and Generate Error Message

The TERROR macro instruction is used to analyze an error associated with a previous TPL-based macro instruction, and to generate an error message describing the error that can be written to a log data set, or displayed to the system operator or local user.

[*symbol*] TERROR [VERBATIM | SUMMARY,] MF = (E, *tpl_address*)

Syntax Description

VERBATIM | SUMMARY	Indicates the type of message to be generated. If the option is VERBATIM, then a literal message is generated that contains the actual values of TPL fields and other information that may be useful in diagnosing the error. If the option is SUMMARY, then the TPL is analyzed and a message is generated that summarizes the error. Default: SUMMARY (generate summary message)
MF = (E, *tpl_address*)	Indicates the execute form of the TERROR macro instruction. The second sublist operand, *tpl_address*, specifies the address of the TPL associated with a request that completed abnormally. Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters. Default: none (must be coded as indicated)

Completion Information

The TERROR macro instruction completes normally when an error message has been formatted, and is ready to be output by the application program. The message is returned in a storage area allocated from subpool 0, and is formatted as a multi-line message compatible with the WTO and WTP macro instructions.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY). Register 0 contains the address of a data structure (TEM) containing the error message. No information is stored in the TPL, and no other information is returned.

If the TERROR macro instruction completes abnormally, no error message is formatted or returned. The TPL or TPL address may be corrupted, and should not be reused. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field is not modified, and contains the information returned by the previous TPL-based macro instruction.

Note The SYNAD or LERAD exit routines are not entered when the TERROR macro instruction completes abnormally.

Return Codes

This table lists the symbolic names for the TERROR return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-18 TERROR Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code
TROKAY	TEM address	n/a
TRFAILED	TATPLERR	n/a
TRFATLFC	func. code	n/a
TRFATLPL	diag. code	n/a
TRFATLAM	diag. code	n/a
TRFATLAP	diag. code	n/a

Usage Information

The TERROR macro instruction is used to generate an error message after a TPL-based macro instruction has completed abnormally. The application program should display the message to the system or network operator, or log it for later inspection.

These types of messages can be generated:

- Indicated by VERBATIM coded in the first positional operand field, contains diagnostic information extracted from the TPL and other API data structures. This message provides a verbatim description of the error.
- Indicated by SUMMARY, and consists of a summary message based on an analysis of the TPL.

The latter is more appropriate for displaying to a non-technical user of the application program.

Note The SUMMARY format is not implemented at this time.

The TERROR macro instruction should be executed after a TPL-based macro instruction completes abnormally with a general return code of 4 (TRFAILED), and a recovery action code less than 24 (TATPLERR). If the recovery action code was TATPLERR, no information was stored in the TPL return code field, and the error message generated is not meaningful. If the general return code was greater than 4 (TRFATLFC, TRFATLPL, TRFATLAM, and TRFATLAP), a fatal error occurred. Issuing the TERROR macro instruction in the latter case probably results in a similar error.

The message is returned in a storage area allocated from subpool 0. The message itself is formatted as a multi-line WTO parameter list. The TEM dsect generated by the TDSECT macro instruction maps the storage area, and may be used to modify fields in the parameter list before a WTO or WTP macro instruction is issued. Alternatively, the application program can extract the message from the parameter list, and output it using whatever means are appropriate. The storage area should be freed when it is no longer required. The first fullword in the storage area contains the subpool and length of the storage area.

Example

This example demonstrates how the TERROR macro instruction is intended to be used. It is assumed that register 1 contains the address of a TPL that has completed abnormally:

```
TERROR VERBATIM,MF=(E,(1))
LTR 15,15
BNZ SKIPWTO
LR 2,0
XR 0,0
USING TEM,2
WTO MF=(E,TEMWTO)
L 0,TEMSL
FREEMAIN R,LV=(0),A=(2)
```

Format of a Verbatim Message

The format of a verbatim message is fixed. This is an example of verbatim message format:

```
T01API001I Transport endpoint error
        JOB jjjjjjjj STEP ssssssss APPL aaaaaaaa USER uuuuuuuu
        TPL xxxxxxxx APCB xxxxxxxx QLSN xxxxxxxx
        SEPM STAT xx TSTAT xx (xxxxxxxx)
        TPL IDENT xx FNCCD xx (ffffffff) ACTIV xx FLAGS xx
        TPL ACTCD xx ERRCD xx (eeeeeeee) DGNCD xxxx (mmmmmmmm)
        TPL EPID xxxxxxxx ECBXR xxxxxxxx OPTCD xxxxxxxx
        TPL PARM1 xxxxxxxx PARM2 xxxxxxxx PARM3 xxxxxxxx
        TPL ADBUF xxxxxxxx DABUF xxxxxxxx OPBUF xxxxxxxx
        TPL ADLEN xxxxxxxx DALEN xxxxxxxx OPLEN xxxxxxxx
```

Upper case fields are generated exactly as shown. Lower case fields are edited from information contained in the API data structures. Edit fields containing lower case *x*'s (for example, xxxxxxxx) represent hexadecimal values. All other fields contains alphanumeric character strings. Each line of the message is described separately:

Table 1-19 describes the Summary Message Format

Table 1-19 Summary Message Format

Line 1	Appears exactly as shown, and identifies the message as being a verbatim error message.
Line 2	Contains the job name (JOB), step name (STEP), application name (APPL), and endpoint user name (USER) associated with the endpoint and TPL.
Line 3	Contains the TPL address (TPL), APCB address (APCB), number of pending connect indications (QLSN).
Line 4	Contains the endpoint state word consisting of the internal state (SEPM STAT) and the current TLI state (TSTAT). ssssssss is the TSW symbolic name for the current state value.
Line 5	Contains TPL fields consisting of the TPL control block identifier (IDENT), the function code (FNCCD), the active semaphore (ACTIV), and various flag bits (FLAGS). ffffffff is the TPL symbolic name for the function code.
Line 6	Contains the TPL return code consisting of the recovery action code (ACTCD), the specific error code (ERRCD), and the diagnostic code (DGNCD). eeeeeeee is the TPL symbolic name for the specific error code, and mmmmmmmm is the module name that generated the error (derived from diagnostic code).

Table 1-19 Summary Message Format (Continued)

Line 7	Contains TPL fields consisting of the endpoint identifier (EPID), the ECB or completion exit routine address (ECBXR), and option codes (OPTCD).
Line 8	Contains the three TPL fixed-length parameters (PARM1, PARM2, and PARM3).
Line 9	Contains the addresses of the three variable-length parameters–the protocol address (ADBUF), user data (DABUF), and protocol options (OPBUF).
Line 10	Contains the length of each variable-length parameter whose address appears in the line above (ADLEN, DALEN, and OPLEN).

The TERROR macro instruction should not be issued after a TOPEN failure. If this is attempted, unpredictable results occur.

TEVNTLST

Create an Event List

The TEVNTLST macro instruction can associate an exit list with an endpoint via a TOPEN macro instruction or allows ECBs to be specified for protocol event and TPEND notifications.

TEVNTLST [,CONNECT = (*address*, ECB | EXIT)]
[,CONFIRM = (*address*, ECB | EXIT)]
[,DATA = (*address*, ECB | EXIT)]
[,XDATA = (*address*, ECB | EXIT)]
[,DGERR = (*address*, ECB | EXIT)]
[,DISCONN = (*address*, ECB | EXIT)]
[,RELEASE = (*address*, ECB | EXIT)]
[,SENDWIND=(*address*, ECB | EXIT)]
[,TPEND = (*address*, ECB | EXIT)]
[,MF = (L | M ([*exit_list_address*])]

Syntax Description

CONNECT = (*address,* ECB | EXIT)
CONFIRM = (*address,* ECB | EXIT)
DATA = (*address,* ECB | EXIT)
XDATA = (*address,* ECB | EXIT)
DGERR = (*address,* ECB | EXIT)
DISCONN = (*address,* ECB | EXIT)
RELEASE = (*address,* ECB | EXIT)
SENDWIND = (*address,* ECB | EXIT)

The first subparameter specifies the address of a routine to be entered or an ECB to be posted when certain asynchronous protocol events occur. If an exit routine is used, the second parameter specifies whether the first subparameter is an ECB or an exit address. The address of a parameter list (mapped by the TXP dsect) is passed to the routine in register 1. An event code (TXPEVENT) stored in the parameter list by the API identifies the event when the same exit routine is used to handle more than one protocol event. Refer to Recovery Action Codes For LERAD Routine Entry: for event codes that are defined.

Default: 0 (no exit routine)

TPEND = (*address,* ECB | EXIT)

The first subparameter specified the address of a routine to be entered or an ECB to be posted when the transport provider terminates and can no longer provide service to the application program. If an exit routine is used, the second parameter specifies whether the first subparameter is an ECB or an exit address. The address of a parameter list (mapped by the TXP dsect) is passed to the routine in register 1. A reason code (TXPREASN) stored in the parameter list by the API identifies the reason for termination of service. Refer to TPEND Reason Codes for reason codes that are defined.

Default: 0 (no TPEND exit routine)

MF = (L | M [*exit_list_address*])

Indicates the list or modify form of the TEVNTLST macro instruction. The second sublist operand, *exit_list_address*, is the address of a storage area that contains (MF=M), or will contain (MF=L), the exit list (TXL). If the exit list address is not provided, or the MF operand is not coded, the exit list is generated in line with the macro instruction.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: not indicated (nonreentrant, in-line list)

Event Codes

This section lists the event codes passed to the CONNECT, CONFIRM, DATA, XDATA, DGERR, DISCONN, RELEASE, and SENDWIND exit routines.

Table 1-20 Event Codes

Name	Dec	Hex	Exit	Protocol Event
TXPECONN	0	X`00'	CONNECT	Connect indication
TXPECONF	4	X`04'	CONFIRM	Confirm indication
TXPEDATA	8	X`08'	DATA	Normal data indication
TXPEXPDT	12	X`0C'	XDATA	Expedited data indication
TXPERROR	16	X`10'	DGERR	Datagram error indication
TXPEDISC	20	X`14'	DISCONN	Disconnect indication
TXPERLSE	24	X`18'	RELEASE	Orderly release indication
TXPESWIN	28	X'12'	SENDWIND	Send Window Opened

TPEND Reason Codes

The following table lists the reason codes for TPEND calls. A reason code (TXPREASN) stored in the parameter list by the API identifies the reason for termination of service.

Table 1-21 TPEND Reason Codes

Name	Dec	Hex	Explanation
TXPRDRAN	0	X`00'	Operator drained subsystem
TXPRSTOP	4	X`04'	Operator stopped subsystem
TXPRTERM	8	X`08'	Subsystem abnormally terminated

Completion Information

If the MF operand is not coded, or MF=L is indicated (without TXL address), the exit list is generated at assembly time, and no executable code is expanded.

Otherwise, the macro instruction expansion contains executable code to generate or modify the exit list at the location specified by the application program. The general return code returned in register 15 is always 0 (TROKAY). No other information is returned.

Return Codes

No error codes are generated.

Usage Information

The TEVNTLST macro instruction builds a list of the address supplied by the TEVNTLST operands. Each address identifies an application program routine to be given control, or ECB to be posted, when the respective event occurs.

The list created by TEVNTLST is referenced by the TOPEN parameter EVENTLST. The structure of the event list is the same as the exit list built by the TOPEN form of TEXLST. An additional set of flags indicate whether each address is the address of an exit routine or an ECB.

Empty slots are created for operands that are not coded on the TEVNTLST macro instruction. Empty slots are set to zero, indicating that no exit routine as been specified. The application program can use the modify form of the TEVNTLST macro instruction to update an event list after it has been created. The event list must be aligned on a fullword boundary.

The address of the event list is provided as a parameter to the TOPEN macro instruction. When the endpoint is opened, the event list is permanently linked to the transport user or endpoint. When an event occurs that may require processing by the application program, the event list linked to the endpoint is checked first to see if an exit routine or ECB is defined. If so, the exit routine is entered to process the event or the respective ECB is posted. Otherwise, the exit list linked to the APCB is checked. If no exit routine or ECB is defined, the occurrence of the event must be detected by some other means (generally via return codes stored in the TPL).

When the APCB or endpoint is opened, a copy of the event list or exit list is saved. Therefore, exit routines and ECBs associated with a transport user or any one of its endpoints cannot be changed once the APCB has been opened, or the endpoint has been created. However, if control of an endpoint is passed to another task or address space, a new event list can be specified via the TOPEN macro instruction that acquires control of the endpoint.

Refer to *Cisco IOS for S/390 Assembler API Concepts* for a detailed discussion of exit routines and a listing of TXL DSECT, which maps the event list.

TEXEC

Execute a Transport Service Parameter List

A Transport Service Parameter List (TPL) that has been initialized, or has been used to make a previous request, can be executed or re-executed using the TEXEC macro instruction. Issuing the TEXEC macro instruction with a function code is functionally equivalent to issuing the macro instruction indicated by the function code.

The TEXEC macro instruction accepts all operands defined for other TPL-based macro instructions, plus those defined in this topic. The precise definition and use of a particular operand depends on the value indicated for FNCCD. Refer to the macro instruction description for the indicated function to determine which operands can be coded, and how they are used.

[*symbol*] TEXEC [EP = *endpoint_id*]
[,ADLEN = *protocol_address_length*]
[,ADBUF = *protocol_address_address*]
[,ADALET = *protocol_address_alet*]
[,DALEN = *user_data_length*]
[,DABUF = *user_data_address*]
[,DAALET = *user_address_alet*]
[,OPLEN = *protocol_options_length*]
[,OPBUF = *protocol_options_address*]
[,OPALET = *protocol_options_alet*]
[,QLSTN = *listen_queue_length*]
[,NEWEP = *new_endpoint_id*]
[,SEQNO = *sequence_number*]
[,USER = *endpoint_userid*]
[,TCB = *task_control_block_address*]
[,ASCB = *address_space_control_block_address*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,TRUNC | NOTRUNC]
[,NEGOT | NONEGOT]
[,BLOCK | NOBLOCK]
[,ASSIGN | USE]
[,LOCAL | REMOTE]
[,PRIMARY | SECNDRY | STATS]
[,DECLARE | VERIFY | QUERY | DEFAULT]
[,TP | API]
[,MORE | NOMORE]
[,NORMAL | EXPEDITE]
[,EOM | NOTEOM]
[,DIRECT | INDIR]
[,ABORT | CLEAR]
[,DELETE | PASS]
[,TUB | ACEE]
[,PLAIN | CIPHER]
[,MBUF | NOMBUF]
[,FULL | NOFULL]
[,TIMEOUT | NOTIMEOUT])]
[,FNCCD = TACCEPT | TADDR | TBIND | TCLEAR |
TCLOSE | TCONFIRM | TCONNECT |
TDISCONN | TINFO | TLISTEN | TOPTION |
TRECV | TRECVERR | TRECVFR | TREJECT |
TRELACK | TRELEASE | TRETRACT | TSEND |

TSENDTO | TUNBIND | TUSER
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (G | E, *tpl_address*)]
[,MF = (I | G | E, [*tpl_address*])]

Syntax Description

FNCCD = *function_code*

Indicates which API function to execute. These are the possible values:
TACCEPT - Accept connection request
TADDR - Get protocol address
TBIND - Bind local protocol address
TCLEAR - Clear disconnect indication
TCLOSE - Close endpoint
TCONFIRM - Receive connect confirmation
TCONNECT - Initiate connection request
TDISCONN - Initiate abortive disconnect
TINFO - Get transport protocol information
TLISTEN - Listen for connect indications
TOPTION - Endpoint option management
TRECV - Receive from connected transport user
TRECVERR - Receive datagram error indication
TRECVFR - Receive a datagram
TREJECT - Reject connection request
TRELACK - Acknowledge orderly release indication
TRELEASE - Initiate or complete orderly release
TRETRACT - Retract a pending TLISTEN request
TSEND - Send to connected transport user
TSENDTO - Send a datagram
TUNBIND - Unbind local protocol address
TUSER - Associate user with endpoint

If a function code is specified, the definition and use of other operands is determined by the designated function. The operands that may be coded, and the rules that apply, are the same as those defined for the API macro instruction that corresponds to the function code. If a function code is not specified, the value stored in the TPL designates the function to be executed.

Default: not indicated (use function code in TPL)

MF = (I | G | E, [*tpl_address*])

Indicates the standard, generate, or execute form of the TEXEC macro instruction. The second sublist operand, tpl_address, is the address of a storage area that contains (MF=E), or contains (MF=G), the Transport Function Parameter List (TPL). If the MF operand is not coded, an in-line list and subroutine linkage is generated.

If the list or modify form is desired, use the TPL macro instruction.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

Completion information is determined by the API function executed, and is defined in the section that describes the macro instruction corresponding to the function executed.

In general, on normal return to the application program, register 15 contains 0 (TROKAY) and register 0 contains a conditional completion code, or 0 (TCOKAY) if there was no conditional completion. On abnormal return, register 15 contains the general return code (unless modified by the SYNAD or LERAD exit routine), and register 0 contains the recovery action code. The recovery action code and a specific error code may also be stored in the return code field of the TPL.

Return Codes

This table lists the symbolic names for the TEXEC return codes. The values associated with the symbolic names can be found in the TPL macro expansion.These return codes are common to all API TPL-based macro instructions:

Note For a description of the return codes that apply to a specific function, refer to the description of the macro instruction that corresponds to the function.

Table 1-22 TEXEC Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY		
TRFAILED	TAENVIRO	TESYSERR TESTOP	TESUBSYS TETERM	TEDRAIN
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD	TEBDECB
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		

Table 1-22 TEXEC Return Codes (Continued)

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.

Usage Information

The TEXEC macro instruction is used to execute a Transport Service Parameter List (TPL) that has already been generated, or to re-execute a TPL that has been previously executed. The TEXEC macro instruction is complementary with the TPL macro instruction that is used to generate or modify a TPL.

The API macro instructions use the same inner macro instructions to generate the macro expansion. Therefore, except for functions performed in the outer most macro instruction, the TEXEC macro instruction used with a function code is functionally equivalent to the macro instruction that corresponds to the particular function.

Example

The macro instruction TEXEC QLSTN=5,FNCCD=TBIND,MF=(E,BINDTPL) generates the same expansion as TBIND QLSTN=5,MF=(E,BINDTPL).

In fact, the TEXEC macro instruction can be used to generate any TPL-based request other than TERROR, TCHECK, TSTATE, and TOPEN.

Most of the function-dependent validity checking is performed by the outer macro instruction (TBIND in the previous example). Therefore, the application programmer should be cautious when using the TEXEC macro instruction to make arbitrary requests.

Example

This macro instruction expands successfully at assembly time, but probably completes abnormally at runtime:

```
TEXEC QLSTN=5,FNCCD=TRECV,MF=(E,BINDTPL)
```

Whereas, this equivalent macro instruction fails at assembly time:

```
TRECV QLSTN=5,MF=(E,BINDTPL)
```

TEXLST

Create an Exit List

The TEXLST macro instruction is used to create an exit list that can be associated with a transport user via an AOPEN macro instruction, or associated with an endpoint via a TOPEN macro instruction.

[*symbol*] TEXLST [AOPEN | TOPEN]
[,SYNAD = *exit_routine_address*]
[,LERAD = *exit_routine_address*]
[,CONNECT = *exit_routine_address*]
[,CONFIRM = *exit_routine_address*]
[,DATA = *exit_routine_address*]
[,XDATA = *exit_routine_address*]
[,DGERR = *exit_routine_address*]
[,DISCONN = *exit_routine_address*]
[,RELEASE = *exit_routine_address*]
[,SENDWIND = *exit_routine_address*]
[,TPEND = *exit_routine_address*]
[,APEND = *exit_routine_address*]
[,MF = (L | M [,*exit_list_address*])]

Syntax Description

AOPEN | TOPEN

Indicates whether the exit list is associated with a transport user or a particular endpoint.

If the option is AOPEN, then the exit list is linked to the APCB with an AOPEN macro instruction.

If the option is TOPEN, then the exit list is linked to an endpoint via a TOPEN macro instruction. Since the exit routines associated with an endpoint are a subset of those defined for the APCB, this operand also determines the maximum length of the exit list.

Default: AOPEN (exit list linked via APCB)

SYNAD = *exit_routine_address*

Indicates the address of a routine to be entered if a physical error or other unusual condition occurs during the processing of a TPL-based request. Invalid requests and logic errors are handled by the LERAD exit routine.

A recovery action code is passed to the exit routine in register 0, and a copy is stored in the return code field of the TPL associated with the request. A specific error code is also stored in the return code field. The address of the TPL is passed in register 1. Refer to Recovery Action Codes for SYNAD Routine Entry for the recovery action codes that cause the SYNAD routine to be entered:

Default: 0 (no SYNAD exit routine)

LERAD = *exit_routine_address*

Indicates the address of a routine to be entered when the application program issues a TPL-based request that results in a logic error. Physical errors or other unusual conditions are handled by the SYNAD exit routine.

A recovery action code is passed to the exit routine in register 0, and a copy is stored in the return code field of the TPL associated with the request. A specific error code is also stored in the return code field. However, if the recovery action code is 24 (TATPLERR), the TPL may be active or corrupted, and no information is stored in the return code field. The address of the TPL is passed in register 1. Refer to Recovery Action Codes For LERAD Routine Entry: for a list of recovery action codes that cause the LERAD routine to be entered:

Default: 0 (no LERAD exit routine)

CONNECT = *exit_routine_address*
CONFIRM = *exit_routine_address*
DATA = *exit_routine_address*
XDATA = *exit_routine_address*
DGERR = *exit_routine_address*
DISCONN = *exit_routine_address*
RELEASE = *exit_routine_address*
SENDWIND = *exit_routine_address*

Indicates the address of a routine to be entered when certain asynchronous protocol events occur. The address of a parameter list (mapped by the TXP dsect) is passed to the routine in register 1. An event code (TXPEVENT) stored in the parameter list by the API identifies the event when the same exit routine is used to handle more than one protocol event. Refer to Recovery Action Codes For LERAD Routine Entry: for event codes that are defined.

Default: 0 (no exit routine)

TPEND = *exit_routine_address*

Indicates the address of a routine to be entered when the transport provider terminates and can no longer provide service to the application program. The address of a parameter list (mapped by the TXP dsect) is passed to the routine in register 1. A reason code (TXPREASN) stored in the parameter list by the API identifies the reason for termination of service. Refer to TPEND Reason Codes for reason codes that are defined.

Default: 0 (no TPEND exit routine)

APEND = *exit_routine_address*

Indicates the address of a routine to be entered when the API subsystem terminates and can no longer provide service to the application program. The address of a parameter list (mapped by the TXP dsect) is passed to the routine in register 1. A reason code (TXPREASN) stored in the parameter list by the API identifies the reason for termination of service. These reason codes are defined:

Default: 0 (no APEND exit routine)

MF = (L | M, [*exit_list_address*])

Indicates the list or modify form of the TEXLST macro instruction. The second sublist operand, exit_list_address, is the address of a storage area that contains (MF=M), or will contain (MF=L), the exit list. If the exit list address is not provided, or the MF operand is not coded, the exit list is generated in line with the macro instruction.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: not indicated (nonreentrant, in-line list)

Recovery Action Codes

This section lists the Recovery action codes for SYNAD and LERAD routine entry. The values associated with the symbolic names can be found in the TPL macro expansion.These return codes are common to all API TPL-based macro instructions:

Table 1-23 Recovery Action Codes for SYNAD Routine Entry

Name	Dec	Hex	Explanation
TAEXCTPN	4	X`04'	Exceptional condition
TAINTEG	8	X`08'	Connection or data integrity error
TAENVIRO	12	X`0C'	Environmental condition

Table 1-24 Recovery Action Codes For LERAD Routine Entry:

Name	Dec	Hex	Explanation
TAFORMAT	16	X`10'	Format or specification error
TAPROCED	20	X`14'	Sequence or procedural error
TATPLERR	24	X`18'	Logic error with no TPL return code

Event Codes

This section lists the event codes passed to CONNECT, CONFIRM, DATA, XDATA, DGERR, DISCONN, RELEASE, and SENDWIND exit routines. The values associated with the symbolic names can be found in the TXP macro expansion.

Table 1-25 Event CodesEvent Codes

Name	Dec	Hex	Exit	Protocol Event
TXPECONN	0	X`00'	CONNECT	Connect indication
TXPECONF	4	X`04'	CONFIRM	Confirm indication
TXPEDATA	8	X`08'	DATA	Normal data indication
TXPEXPDT	12	X`0C'	XDATA	Expedited data indication
TXPERROR	16	X`10'	DGERR	Datagram error indication
TXPEDISC	20	X`14'	DISCONN	Disconnect indication
TXPERLSE	24	X`18'	RELEASE	Orderly release indication

Table 1-25 Event CodesEvent Codes (Continued)

Name	Dec	Hex	Exit	Protocol Event
TXPESWND	28	X'18'	SENDWIND	Send Window opened

APEND and TPEND Reason Codes

The following tables list the reason codes for TPEND and APEND calls. A reason code (TXPREASN) stored in the parameter list by the API identifies the reason for termination of service.

Table 1-26 TPEND Reason Codes

Name	Dec	Hex	Explanation
TXPRDRAN	0	X`00'	Operator drained subsystem
TXPRSTOP	4	X`04'	Operator stopped subsystem
TXPRTERM	8	X`08'	Subsystem abnormally terminated

Table 1-27 APEND Reason Codes

Name	Dec	Hex	Explanation
TXPRDRAN	0	X`00'	Operator drained subsystem
TXPRSTOP	4	X`04'	Operator stopped subsystem
TXPRTERM	8	X`08'	Subsystem abnormally terminated

Completion Information

If the MF operand is not coded, or MF=L is indicated (without exit_list_address (TXL) address), the exit list is generated at assembly time, and no executable code is expanded.

Otherwise, the macro instruction expansion contains executable code to generate or modify the exit list at the location specified by the application program. The general return code returned in register 15 is always 0 (TROKAY). No other information is returned.

Return Codes

No error codes are generated.

Usage Information

The TEXLST macro instruction builds a list of exit routine addresses. Each operand in this macro instruction represents a class of events for which an exit routine can be entered by the API.

The address supplied for each operand identifies an application program routine to be given control when a particular event occurs.

Example

The SYNAD operand supplies the address of a routine that handles exceptional or unusual conditions (other than logic errors) for TPL-based macro instructions, and the CONNECT operand supplies the address of a routine that receives connect indications.

The length of the exit list depends on whether it is used with the AOPEN or TOPEN macro instruction. Some exit routines defined in the AOPEN exit list cannot be specified in a TOPEN exit list. Therefore, an AOPEN exit list is longer than a TOPEN exit list. A length parameter generated at the beginning of an exit list indicates the type of exit list, and is validity checked at execution time when referenced with an AOPEN or TOPEN macro instruction.

Empty slots are created for operands that are not coded on the TEXLST macro instruction. Empty slots are set to zero, indicating that no exit routine as been specified. The application program can use the modify form of the TEXLST macro instruction to update an exit list after it has been created. The exit list must be aligned on a fullword boundary.

The address of the exit list is provided as a parameter to the AOPEN or TOPEN macro instruction. When the APCB or endpoint is opened, the exit list is permanently linked to the transport user or endpoint. When an event occurs that may require processing by the application program, the exit list linked to the endpoint is checked first to see if an exit routine is defined. If so, the exit routine is entered to process the event. Otherwise, the exit list linked to the APCB is checked. If no exit routine is defined, the occurrence of the event must be detected by some other means (generally via return codes stored in the TPL).

When the APCB or endpoint is opened, a copy of the exit list is saved. Therefore, exit routines associated with a transport user or any one of its endpoints cannot be changed once the APCB has been opened, or the endpoint has been created. However, if control of an endpoint is passed to another task or address space, a new exit list can be specified via the TOPEN macro instruction that acquires control of the endpoint.

Refer to *Cisco IOS for S/390 Planning Guide* for a detailed discussion of exit routines. The exit list is mapped by the TXL DSECT, which is also listed in Appendix D, Data Structures, of the *Cisco IOS for S/390 Assembler API Macros* manual.

TINFO

Retrieve Transport Protocol Information

Protocol information associated with an endpoint and maintained by the transport provider can be retrieved using the TINFO macro instruction.

[*symbol*] TINFO [EP = *endpoint_id*]
[,DALEN = *protocol_information_length*]
[,DABUF = *protocol_information_address*]
[,DAALET = *protocol_information_alet*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,TRUNC | NOTRUNC]
[,PRIMARY | SECNDRY | STATS])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TINFO macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

DALEN = *protocol_information_length*

Indicates the length (in bytes) of the protocol information storage area identified by the DABUF operand. The length is updated when the request is completed to reflect the actual length of protocol information returned. If the length is zero, the API macro instruction completes abnormally.

Default: 0 (return no protocol information)

DABUF = *protocol_information_address*

Indicates the address of a storage area for returning protocol information maintained for the designated endpoint. The storage area should be large enough to contain the requested information, and can be aligned on any boundary convenient to the application program.

The type of information requested is indicated by the OPTCD operand. Only the information indicated by OPTCD=PRIMARY is standardized for all transport providers. All other information types are provider-dependent. The information provided with a OPTCD=PRIMARY request can be used to determine the maximum size of the information unit returned when designating other information types.

Default: (no protocol information storage area)

DAALET = *protocol_information_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the DABUF parameter. The DAALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The DAALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TINFO macro instruction.

If the option is OPTCD=SYNC, then the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, then the request is executed in asynchronous mode, and control is returned immediately after scheduling the TINFO request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = TRUNC | NOTRUNC

Indicates whether the protocol information returned to the application program by the transport provider should be truncated if it does not fit within the storage area provided.

If the option is OPTCD=TRUNC, then the excess is truncated, and the TINFO macro instruction is completed conditionally as long as no other errors occur.

If the option is OPTCD=NOTRUNC, then nothing is placed in the storage area, and the TINFO macro instruction is completed abnormally.

Default: NOTRUNC (no truncation)

OPTCD = PRIMARY | SECNDRY | STATS

Indicates the type of information requested.

PRIMARY - Designates primary protocol information whose format and meaning is standardized for all transport providers. The application program can use this information to determine the basic characteristics of the transport service and limits of the transport provider.

SECNDRY - Designates secondary protocol information whose format and meaning is specific to the transport service being used. This information includes internal protocol and state variables that govern the operation of the transport protocol. Transport providers are not required to support this option code. Transport providers in the current implementation do not support SECNDRY.

STATS - Designates statistical information recorded by the transport provider whose format and meaning is specific to the transport service being used. Transport providers are not required to support this option code. Transport providers in the current implementation do not support STATS.

Default: PRIMARY (return basic protocol information)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TINFO macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TINFO macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TINFO macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TINFO macro instruction completes normally (or conditionally) when the requested protocol information as been returned in the storage area provided by the application program. The length of the storage area is updated to reflect the actual amount of information returned. The state of the endpoint is unchanged.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY), and a conditional completion code is returned in register 0. TCTRUNC is set if the protocol information returned to the application program was truncated to fit in the storage area provided. The TPL return code field is set accordingly. No other information is returned.

If the TINFO macro instruction completes abnormally, no protocol information is returned to the application program. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TINFO return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-28 TINFO Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY	TCTRUNC	
TRFAILED	TAINTEG	TEOVRFLO		
	TAENVIRO	TESYSERR TESTOP	TESUBSYS TETERM	TEDRAIN TEUNSUPO
	TAFORMAT	TEBDFNCD TEBDEXIT	EBDOPCD TEBDDATA	TEBDECB
	TAPROCED	TEAMODE	TEINCMPL	
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TINFO macro instruction is used to obtain information from the transport provider that is maintained for a specific endpoint. An option code is used to indicate what type of information is desired. The information types range from basic protocol information that is common to all transport providers, to detailed protocol information and statistics that are provider-dependent

The basic protocol information is standardized for all transport providers, and is requested by indicating PRIMARY with the OPTCD operand. The information returned can be used by the application program to determine basic characteristics and limits of the transport service.

Maximum lengths of protocol addresses, connect and disconnect user data, and protocol options are provided.

If storage areas used for returning such information by other macro instructions are allocated dynamically using this information, an overflow condition should not occur.

Basic Protocol Information Returned

The basic protocol information returned is fixed in length (76 bytes), and is mapped by the Transport Information Block (TIB) dsect. This table defines the information returned:

Note Each field is designated by the label defined in the TIB dsect that should be used to access it.

TIBTSDOM — A one-byte value that defines the communications domain within which the endpoint was created. The value corresponds to the communications domain requested by the TOPEN macro instruction. If none was explicitly requested, this value indicates the domain that was assigned based on system and installation default values.

TIBTSTYP — A one-byte value that defines the mode of service assigned for the endpoint. The value corresponds to the mode of service requested by the TOPEN service function. If none was explicitly requested, this value indicates the mode of service that was assigned based on system and installation default values.

TIBTSCHR — A flag byte defining one of these various characteristics of the transport service:

- Message boundaries are preserved within data stream
- Expedited data is supported
- The transport provider supports the protocol options parameter
- User data can be sent with connection request
- User data can be sent with disconnect request

TIBTSOPT — A flag byte indicating one of these additional options supported by the transport provider:

- Datagram address associations
- Orderly release
- Secondary protocol information
- Statistical information

TIBSYSID	Contains the four-character ID of the MVS subsystem containing the transport service provider (or its surrogate). The value is the same as the subsystem ID requested by the AOPEN service function. If none was explicitly requested, this value is the ID of the MVS subsystem that was assigned based on system and installation default values.
TIBSVCID	Contains the eight 8-character ID of the transport service provider that is managing the endpoint. The value is the same as the transport service ID requested by the TOPEN service function. If none was provided, this value is the ID of the transport service provider that was assigned based on system and installation default values.
TIBPROTO	Contains the protocol number of the protocol used to provide the transport service. The value is the same as the protocol number requested by the TOPEN service function. If none was provided, this value is the protocol number of the protocol that was assigned based on system and installation default values.

Transport Service Limits

The API uses a general notation for defining limits of the transport service. The limit of a particular facility is represented by a signed, integer value.

- If the value is greater than zero, the facility is supported by the transport provider, and the value is the limit of the facility.
- If the value is -1, the facility is supported, but there is no limit.
- If the value is - 2, the facility is not supported at all.

A value of zero is sometimes used to represent special characteristics of the facility.

Transport Interface Limits

Table 1-29 describes the Fullword Values that Define Limits of the Transport Interface

Table 1-29 Fullword Values that Define Limits of the Transport Interface:

TIBQLSTN	A value greater than zero indicates the maximum number of connect indications that can be queued by the transport interface. A value of -1 indicates that there is no limit on the size of the connect indication queue, and a value of -2 specifies that the transport interface does not support the queueing of connect indications. A value of zero is not returned.
TIBQSEND	A value greater than zero indicates the maximum number of uncompleted send requests that can be queued by the transport interface. A value of -1 indicates that there is no limit on the number of send requests. Since the transport interface must allow at least one uncompleted send request, a value of -2 or zero is never returned.
TIBQRECV	A value greater than zero indicates the maximum number of uncompleted receive requests that can be queued by the transport interface. A value of -1 indicates that there is no limit on the number of receive requests. Since the transport interface must allow at least one uncompleted receive request, a value of -2 or zero is never returned.

Table 1-29 Fullword Values that Define Limits of the Transport Interface: (Continued)

TIBLTSND	A value greater than zero indicates the maximum number of user data bytes that can be transferred by the transport interface with a single send request. A value of -1 indicates that there is no limit on the amount of data in a single send request. The transport interface always supports the sending of data, and a value of -2 or zero is never returned.
TIBLTRCV	A value greater than zero indicates the maximum number of user data bytes that can be transferred by the transport interface with a single receive request. A value of -1 indicates that there is no limit on the amount of data in a single receive request. The transport interface always supports the receiving of data, and a value of -2 or zero is never returned.
TIBLSEND	A value greater than zero indicates the maximum number of user data bytes that can be pending for uncompleted send requests. A value of -1 indicates that there is no limit on the total amount of pending send data. A value of -2 or zero is never returned.
TIBLRECV	A value greater than zero indicates the maximum number of user data bytes that can be pending for uncompleted receive requests. A value of -1 indicates that there is no limit on the total amount of pending receive data. A value of -2 or zero is never returned.

Transport Provider Limits

Table 1-30 describes the Fullword Values that Define Limits of the Transport Provider

Table 1-30 Fullword Values that Define Limits of the Transport Provider

TIBLADDR	A value greater than zero indicates the maximum size of a transport protocol address. A value of -1 indicates that there is no limit on the address size, and a value of -2 indicates that the transport provider does not provide user access to transport protocol addresses. A value of 0 is not returned by the transport provider.
TIBLOPTN	A value greater than zero indicates the maximum number of bytes in the protocol options parameter supported by the transport provider. A value of -1 indicates that there is no limit on the size of protocol options, and a value of -2 specifies that the transport provider does not support user-specified protocol options. A value of 0 is not returned by the transport provider.
TIBLTSDU	A value greater than zero indicates the maximum size of a Transport Service Data Unit (TSDU). A value of -1 indicates that there is no limit on the size of a TSDU, and a value of -2 indicates that the transfer of normal data is not supported by the transport provider. A value of zero indicates that the transport provider does not support the concept of a TSDU, although it does support the sending of a data stream with no logical boundaries preserved across the connection.
TIBLXPDT	A value greater than zero indicates the maximum size of an Expedited Transport Service Data Unit (ETSDU). A value of -1 indicates that there is no limit on the size of an ETSDU, and a value of -2 indicates that the transfer of expedited data is not supported by the transport provider. A value of zero indicates that the transport provider does not support the concept of an ETSDU, although it does support the sending of an expedited data stream with no logical boundaries preserved across a connection.

Table 1-30 Fullword Values that Define Limits of the Transport Provider (Continued)

TIBLCONN	A value greater than zero indicates the maximum number of bytes of user data that can be transferred during connection establishment. A value of -1 indicates that there is no limit on the amount of user data that can be transferred, and a value of -2 specifies that the transport provider does not support connect user data. A value of 0 is not returned by the transport provider.
TIBLDISC	A value greater than zero indicates the maximum number of bytes of user data that can be transferred during connection release. A value of -1 indicates that there is no limit on the amount of user data that can be transferred, and a value of -2 specifies that the transport provider does not support disconnect user data. A value of 0 is not returned by the transport provider.
TIBLINFO	A value greater than zero indicates the maximum size of an information unit returned by the TINFO service function for information types other than PRIMARY. A value of -1 indicates that there is no limit on the size of an information unit, and a value of -2 indicates that the transport provider does not support any information types other than PRIMARY. A value of 0 is not returned by the transport provider.

An application program can minimize its dependence on a particular transport provider by using the information defined in the previous tables to determine which facilities and options are supported, and the maximum size of variable-length storage areas used by macro instructions The content and format of the remaining information types is provider-dependent. Only PRIMARY information is supported by the transport providers in the current implementation.

TLISTEN

Listen for a Connect Indication

The TLISTEN macro instruction is used to listen for connect indications generated by connection requests arriving at an endpoint operating in server mode. The endpoint must have been previously enabled with a TBIND macro instruction. The protocol address of the remote transport user that initiated the connection request is returned by the transport provider.

[*symbol*] TLISTEN [EP = *endpoint_id*]
[,ADLEN = *protocol_address_length*]
[,ADBUF = *protocol_address_address*]
[,ADALET = *protocol_address_alet*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,TRUNC | NOTRUNC]
[,BLOCK | NOBLOCK])]
[,ECB = INTERNAL | *event_control_block_addr*]
[EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TLISTEN macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

ADLEN = *protocol_address_length*

Indicates the length (in bytes) of the protocol address storage area identified by the ADBUF operand. The length is updated when the request is completed to reflect the actual length of the protocol address returned. If the length is zero, the protocol address of the calling transport user is not returned to the application program.

Default: 0 (return no protocol address)

ADBUF = *protocol_address_address*

Indicates the address of a storage area for returning the protocol address of the calling transport user. The storage area should be large enough to contain the entire address. The format of the protocol address is provider-dependent, and its maximum size can be determined by issuing a TINFO macro instruction. The storage area can be aligned on any boundary.

Default: 0 (no protocol address storage area)

ADALET = *protocol_address_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the ADBUF parameter. The ADALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The ADALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TADDR macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TLISTEN request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = TRUNC | NOTRUNC

Indicates whether the information returned to the application program by the transport provider should be truncated if it does not fit within the storage area provided.

If the option is OPTCD=TRUNC, the excess is truncated, and the TLISTEN macro instruction is completed conditionally as long as no other errors occur.

If the option is OPTCD=NOTRUNC, nothing is placed in the storage area, and the TLISTEN macro instruction is completed abnormally.

Default: NOTRUNC (no truncation)

OPTCD = BLOCK | NOBLOCK

Indicates whether or not the issuing task can be suspended if the TLISTEN macro instruction cannot be completed immediately.

If the option is OPTCD=BLOCK (and no connect indicated has been generated), the issuing task is suspended until a connection request arrives.

If the option is OPTCD=NOBLOCK, the macro instruction is completed immediately, and an abnormal return code indicates that the task would have been suspended for an indefinite period of time.

The TLISTEN macro instruction can be used to poll for new connect indications. If a connect indication is available, the request is completed as usual. Otherwise, the request is completed abnormally and the transport user can try again after delaying an appropriate period of time.

In either case, if a connect indication has already been generated, the TLISTEN macro instruction completes normally without suspending the issuing task.

Default: BLOCK (suspend issuing task if necessary)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TLISTEN macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TLISTEN macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TLISTEN macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TLISTEN macro instruction completes normally (or conditionally) when a connect indication is pending, and the information accompanying the connection request has been returned to the application program. If the number of pending indications is one, the state of the endpoint is changed from enabled (TSENABLD) to connect-indication-pending (TSINCONN). Otherwise, connect indications were already pending, and the state is not changed.

If a storage area was provided by the application program, the protocol address of the calling transport user is returned. The corresponding length of the address buffer storage area is updated to reflect the actual amount of information returned.

A sequence number that identifies the connect indication is returned in the TPLSEQNO field of the TPL associated with this request. If more connect indications have been generated, and are waiting to be received by the application program, TOMORE is set in TPLOPCD2, and the number of available connect indications is returned in the TPLCOUNT field of the TPL.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY), and a conditional completion code is returned in register 0. TCTRUNC is set if the information returned to the application program was truncated to fit in the storage area provided. The TPL return code field is set accordingly. No other information is returned.

If the TLISTEN macro instruction completes abnormally, no information is returned, and the connect indication (if any) remains available. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TLISTEN return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-31 TLISTEN Return Codes:

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY	TCTRUNC	
TRFAILED	TAEXCPTN	TENOBLOK		
	TAINTEG	TEPROTO	TEOVRFLO	TERETRCT
	TAENVIRO	TESYSERR TESTOP	TESUBSYS TETERM	TEDRAIN TEUNSUPF
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD TEBDADDR	TEBDECB
	TAPROCED	TEAMODE	TESTATE	TEINCMPL

Table 1-31 TLISTEN Return Codes: (Continued)

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation
	TATPLERR	TEACTIVE
TRFATLFC	func. code	The function code loaded into register 0 was invalid.
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.

Usage Information

The TLISTEN macro instruction is used to listen for connect indications that are generated at an endpoint. A connect indication is generated when a connection request arrives from a remote transport user that requests a connection to the local endpoint. The indication remains pending until accepted or rejected by the application program, or retracted by the remote transport user.

The TLISTEN macro instruction completes when a connect indication is available. A sequence number is returned in the TPLSEQNO field of the TPL associated with the request to uniquely identify the pending indication. This sequence number must be provided with subsequent macro instructions that accept (TACCEPT) or reject (TREJECT) the connection request. The application program should make no assumptions regarding the format of the sequence number other than it is an unsigned fullword value. In particular, sequential completions of TLISTEN requests do not necessarily generate sequential sequence numbers. Also, the sequence number is not necessarily a small integer value suitable for array indexing.

A count of the number of unreceived connect indications is returned in the TPLCOUNT field of the TPL. If the value returned is non-zero, another TLISTEN macro instruction should be issued to receive the next connect indication. The API guarantees that a TLISTEN request is completed immediately if the preceding TLISTEN completed with a non-zero indication count. TOMORE is also set in TPLOPCD2 to indicate more connect indications are available. TOMORE corresponds to the option code that is set when OPTCD=MORE is indicated.

The protocol address of the calling transport user is also returned to the application program. The application program can use the protocol address to determine if it should accept or reject the connection request.

The TLISTEN macro instruction is generally used by application programs running in server mode. The protocol address bound to the endpoint is well known by transport users that want to connect to the application program and use its services. Connect indications generated by arriving connection requests are queued, and are presented to the application program as TLISTEN macro instructions are issued. The total number of connect indications that can be queued at one time is determined by the TBIND macro instruction.

The TLISTEN macro instruction does not remove a connect indication from the queue, but only serves to retrieve the information associated with the indication. When a connect indication has been received by the application program, it is said to be pending. A TACCEPT or TREJECT macro instruction must be issued to remove the pending indication from the queue. More than one connect indication can be received with TLISTEN macro instructions before any are accepted or rejected, and pending indications can be accepted or rejected in any order.

The number of transport users that can be connected at one time is controlled by the number of endpoints the application program is able to create. However, the number of connection requests that can be awaiting acceptance is controlled by the maximum length of the connect indication queue that was specified when the endpoint was enabled. If this length is zero, the endpoint is disabled, and cannot queue any connect indications. An error is generated if a TLISTEN macro instruction is issued at an endpoint that is disabled.

If OPTCD=BLOCK is indicated when the TLISTEN macro instruction is issued, the request is not completed until a connect indication is available. If the application needs to use the endpoint for some other purpose, an outstanding TLISTEN request must be retracted. The TRETRACT macro instruction causes a pending TLISTEN to complete immediately with a return code indicating the retraction.

The TLISTEN macro instruction is normally issued at endpoints operating in connection mode. However, if an endpoint operating in connectionless mode was enabled for (simulated) connect indications (see TBIND), the TLISTEN macro instruction should be used to receive connect indications generated by arriving datagrams. A subsequent TACCEPT macro instruction creates an association with the remote transport user. See *Cisco IOS for S/390 Assembler API Concepts* for a discussion of associations in connectionless mode.

TOPEN

Open a Transport Endpoint

The TOPEN macro instruction is used to create an endpoint within a given communications domain, and to designate the type of transport service required for the endpoint. Optionally, the TOPEN macro instruction can be used to acquire control of an existing endpoint from another endpoint ID created by another task or in another address space.

[*symbol*] TOPEN [DOMAIN = INET]
[,TYPE = (*mode* [,*options*])]
[,PROTO = *protocol_number*]
[,SVCID = *transport_service_id*]
[,APCB = *application_program_control_block_addr*]
[,EXLST = *exit_list_address*]
[,EVENTLST = *event_list_address*]
[,UCNTX = *one_word_of_user_context*]
[,EP = *old_endpoint_id*]
[,TCB = *task_control_block_address*]
[,ASCB = *address_space_control_block_address*]
[,USER = *endpoint_userid*]
[,MODE = TLI | SOCKETS]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,TUB | ACEE]
[,PLAIN | CIPHER]
[,NEW | OLD])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tspl_address*])]

Syntax Description

DOMAIN = INET

Indicates the communications domain within which the new endpoint exists. Once a domain has been selected and the endpoint has been created, the domain to which it belongs cannot be changed for the life of the endpoint. If a domain is not specified, one is selected based on the specification of other parameters and installation defaults.

Definition of a domain is independent of the existence of a transport provider on the local system that supports the domain. If a domain is selected for which no transport provider exists, the endpoint is not created. In the current implementation, only a transport provider for DOMAIN=INET is supported

Default: INET

TYPE = (*mode* [,*options*])

Indicates the type of transport service required for the endpoint. The service type in combination with the domain specification is generally sufficient to determine the protocol and transport provider that are used to provide the service. If the transport service type is not specified, one is selected based on the specification of other parameters and installation defaults.

The service type is fixed for the life of the endpoint. The service type is a sublist consisting of the mode of service, followed by optional services required by the endpoint.

The mode of service must be one of these, and must be coded as the first sublist operand:

COTS - Connection-mode transport service

CLTS - Connectionless-mode transport service

Two optional services are defined, and can be requested by specifying one of these keywords:

ORDREL - Orderly release required. ORDREL should be requested only with the COTS service mode.

ASSOC - Connectionless associations required. ASSOC should be requested only with the CLTS service mode.

Since the association service is supported entirely within the transport service interface, orderly release is always available when connectionless-mode associations are used.

These combinations are valid types of service:

```
COTS   (COTS,ORDREL)
CLTS   (CLTS,ASSOC)
```

Domains do not necessarily support all transport service types, and for any given service type that is supported by a particular domain, a transport provider may not be available on the local system. If no transport provider exists for the specified service type, the endpoint is not created. This operand is mutually exclusive with the PROTO operand.

Default: not indicated (use installation default)

PROTO = *protocol_number*

Indicates the protocol number of the transport protocol within a communication domain that is required for the endpoint. The protocol number in combination with the domain specification is generally sufficient to determine the protocol and transport provider that are used to provide the service. If the protocol number is not specified, one is selected based on the specification of other parameters and installation defaults. The protocol number is fixed for the life of the endpoint.

Generally, the transport service type should be used to designate the required protocol service. However, if more than one candidate protocol exits for a given service type, the specific choice can be indicated with the protocol number. Also, some installations may be able to run production and development versions of the same protocol service, and in this case, a protocol number must be specified to force selection of the development version.

The protocol number is domain-dependent, and identifies a specific protocol. This operand is mutually exclusive with the TYPE operand.

Default: not indicated (use installation default)

SVCID = *transport_service_id*

Indicates the ID of a specific transport service provider. Generally, the transport service type or protocol number is sufficient for selecting the appropriate transport service provider. However, if more than one transport service provider is available that can provide the requested service, the transport service ID must be indicated to select the appropriate provider. The transport service ID is coded as an alphanumeric string up to eight (8) characters in length.

Default: not indicated (use installation default)

APCB = *application_program_control_block_address*

Indicates the address of the APCB that defines the application program and corresponding transport user. The APCB also defines the MVS subsystem that contains the transport provider. The APCB must have been opened by the same task that issued the TOPEN macro instruction. If the APCB has not been opened, unpredictable results may occur.

Default: 0 (no APCB address)

EXLST = *exit_list_address*

Links the endpoint with an exit list containing addresses of routines to be entered when certain protocol events occur. This list is created by a TEXLST macro instruction. More than one endpoint can be linked to the same exit list.

If no exit list is provided, the application program is not able to receive immediate notification of asynchronous protocol events. However, notification may still be received synchronously with the completion status that is returned to the application program when a macro instruction completes. The TPEND asynchronous exit, and the SYNAD and LERAD synchronous exits, are specified in the AOPEN exit list, and cannot be defined in the TOPEN exit list. For more information on exit lists, read the TEXLST macro and *Cisco IOS for S/390 Assembler API Concepts*, which discusses synchronization and exit routines.

Note: This operand is mutually exclusive with the following EVENTLST operand.

Default: 0 (no exit list)

EVENTLST = *event_list_address*

Defines a list of ECBs and exits to use for protocol and/or shutdown event notification. The function of the event list is identical to the function of the exit list (EXLST parameter), except that an event list supports event notification via ECB posting as well as exit routine execution. The event list is created using the TEVNTLST macro instruction.

Note: This operand is mutually exclusive with the previous EXLST operand.

Default: 0 (no event list)

UCNTX = *one_word_of_user_context*

Specifies one arbitrary word of user context to be associated with the endpoint. The information provided is not interpreted by the API, and is merely saved with other endpoint information. It may be retrieved later by the application program, and is useful for obtaining context within exit routines. This word of context is included in the parameter list passed to any asynchronous exit routine that is entered on behalf of the endpoint.

Default: 0 (no user context)

EP = *old_endpoint_id*

Specifies the endpoint that is being acquired when OPTCD=OLD is indicated. The value specified must be the endpoint ID of an existing endpoint as returned from a TOPEN macro instruction issued by the controlling task or address space. The task relinquishing control of the endpoint must also issue a TCLOSE OPTCD=PASS macro instruction specifying the same endpoint ID.

Default: 0 (create new endpoint)

TCB = *task_control_block_address*

Indicates the TCB address of the task from which an endpoint is being acquired when OPTCD=OLD is specified.

If the indicated value is zero, the endpoint is acquired from another task in the specified address space that indicates this task's TCB address on a TCLOSE TCB parameter.

If the indicated value is not zero, the value specified must match the TCB address of the task relinquishing control of the endpoint.

Default: 0 (accept from any task)

ASCB = *address_space_control_block_address*

Indicates the ASCB address of the address space from which an endpoint is being acquired when OPTCD=OLD is specified.

If the indicated value is zero, the endpoint can only be acquired from a task executing within the same address space

If the indicated value is not zero, the indicated value is the ASCB address of another address space that currently controls the endpoint

The relinquishing address space must issue a TCLOSE macro instruction indicating this address space as the acquirer.

Default: 0 (accept from task within this address space only)

USER = *endpoint_userid*

Associates a user ID with the endpoint for authorization and accounting purposes.

If the option is OPTCD=TUB, the specified value must be the address of a Transport Endpoint User Block (TUB) containing the user information.

If the option is OPTCD=ACEE, the specified value must be the address of an Accessor Environment Element (ACEE) obtained from the local security system when the user ID was authenticated.

If the option is not coded, the application name specified in the APCB is used.

The password contained in the TUB may be plain text or cipher text depending on the OPTCD=PLAIN | CIPHER operand. If cipher text, it is assumed that the password was encrypted using the encryption mechanism supplied by the local security system. The API merely provides the password to the security system in its encrypted form.

The user ID or application name is also supplied to the transport provider. How this information is used is unspecified, and provider-dependent.

Default: 0 (no user ID; use application name for accounting and authorization)

MODE = TLI | SOCKETS

Allows TSEND and TSENDTO operate in a mode similar to BSD sockets

If the option is MODE=SOCKETS, then data transfer will occur in socket mode.

See Usage Notes for TSEND and TSENDTO for additional information on data transfer modes.

Default: TLI

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = TUB | ACEE

Indicates the format of user ID information referenced by the USER operand.

If the option is OPTCD=TUB, then user ID, group, and password information are provided in a Transport User Block (TUB).

If the option is OPTCD=ACEE, the user information is contained in an Accessor Environment Element (ACEE) obtained from the local security system.

Default: TUB (user information provided in TUB)

OPTCD = PLAIN | CIPHER

Indicates whether the password contained in the Transport User Block (TUB) designated with the USER operand has been encrypted, or is in its plain text form.

If the option is OPTCD=PLAIN, the password is in plain text.

If the option is OPTCD=CIPHER, the password is encrypted.

The API uses this information when requesting user ID and password verification from the local security system.

Default: PLAIN (password in plain text)

OPTCD = NEW | OLD

Indicates whether a new endpoint is to be created, or an existing endpoint is to be passed to another task or address space.

If the option is OPTCD=NEW, a new endpoint is to be created, and the EP operand must indicate a zero value.

If the option is OPTCD=OLD, control of an existing endpoint is being acquired from another task or address space. The EP operand indicates the ID of an existing endpoint.

Default: NEW (create new endpoint)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TOPEN macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TOPEN macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TOPEN macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TOPEN macro instruction completes normally when the requested endpoint has been created (or acquired), and is ready to be used as the argument of other API macro instructions. The initial state of the endpoint is opened (TSOPENED) if the endpoint is new. If the endpoint is old, and has been acquired from another task or address space, the endpoint retains the state that existed when it was closed by the relinquishing task. In this case, the state may be opened (TSOPENED), disabled (TSDSABLD), enabled (TSENABLD), or connected (TSCONNCT).

A token that identifies the endpoint is returned in the TPL as the endpoint ID, and should be used in all subsequent requests that refer to this endpoint. The application program should make no assumptions regarding the format of an endpoint ID, other than it is an unsigned, fullword value. If an existing endpoint was acquired from another task, the TCB and ASCB addresses of the relinquishing task are returned.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY). The conditional completion code in register 0 is always 0 (TCOKAY), and the TPL return code field is set accordingly. No other information is returned.

If the TOPEN function completes abnormally, the endpoint is not created (or acquired), and no endpoint ID is returned. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TOPEN return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-32 TOPEN Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY		
TRFAILED	TAENVIRO	TESYSERR TESTOP TEUNSUPF	TESUBSYS TETERM TEUNAUTH	TEDRAIN TEUNAVBL TERSOURC
	TAFORMAT	TEBDFNCD TEBDEXIT TEBDPROT TEBDUSER TEBDOPCD	TEBDDOM TEBDEPID TEBDACEE TEBDDECB TEBDTYPE	TEBDXLST TEBDTSID TEBADDR TEBDASCB TEBDXECB
	TAPROCED	TEAMODE	TEOWNER	
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TOPEN macro instruction is used to create a new transport endpoint, or acquire control of an existing endpoint from some other task or address space. An endpoint ID is returned that is used to identify the newly created (or acquired) endpoint, and must be supplied in all subsequent transport service requests that apply to the endpoint. The endpoint remains opened until closed with a TCLOSE macro instruction.

The endpoint carries with it some context that is fixed for the duration of its use. It belongs to a particular communications domain, and is associated with a particular transport provider active on the local system. The transport provider is the supplier of a transport service using a particular transport protocol with well known characteristics. The domain, transport service and protocol are selected in accordance with the DOMAIN, TYPE, and PROTO operands specified on the TOPEN macro instruction. A service ID can also be specified which selects a particular provider when more than one apply.

The endpoint is linked to the application program via the APCB address that is specified when the endpoint is opened. The APCB also serves to identify the MVS subsystem that contains the transport provider. Once the endpoint has been opened, the endpoint ID serves as the anchor for all context related to the endpoint. The APCB must have been opened by the same task that opens the endpoint.

The endpoint can also be associated with a user that is known to the local security system. The user is identified with a Transport User Block (TUB), or if the user ID has already been authenticated by the application program, the address of an ACEE can be supplied. This information is used to determine access authority for the requested service, and is used in subsequent requests to determine access authority for certain resources such as well-known protocol addresses. Any accounting information maintained by the API or the transport provider also contains the user ID.

Throughout the life of the endpoint, several asynchronous protocol events can occur. For example, an endpoint used to listen for connection requests can receive a connect indication, or an endpoint associated with an established connection can suddenly become disconnected. An exit list, generated by the TEXLST or TEVNTLST macro instruction, is used to designate exit routines for handling asynchronous events. The address of the exit list is specified by the EXLST or EVENTLST operand.

The exit list is linked to the endpoint at the time it is created, and if no exit list is specified, the exit list linked to the APCB is used in its place. If no exit list exists, or a particular protocol exit has not been enabled, the corresponding event must be processed synchronously.

An endpoint can only be closed by the task that opened it. The opening task is said to control (or own) the endpoint, although other tasks may issue macro instructions that reference the endpoint. If it is necessary for another task to close an endpoint, control must be acquired by the closing task. The current owner passes control by closing the endpoint with OPTCD=PASS indicated. The new owner acquires control by opening the endpoint with OPTCD=OLD indicated. Ownership can be passed to a task in another address space. The task and address space are identified by the TCB and ASCB operands.

When control of an endpoint is passed to another address space, local endpoint context must be recreated in the new address space. Also, since local storage used to maintain the context is associated with the task that allocates it, this context must be recreated, even when passing control to another task within the same address space. Therefore, even though a passed endpoint retains most of its existing context, a new endpoint ID is assigned. The application program must use the endpoint ID returned by TOPEN in all future service requests, and the old endpoint ID should be discarded. Neither the acquiring or relinquishing tasks should ever reference the old endpoint ID once the TOPEN and TCLOSE macro instructions have completed.

TOPTION

Manage Options for Transport Endpoint

Protocol options associated with an endpoint are managed using the TOPTION macro instruction. Options can be declared, queried or verified, and default options used by the transport provider can be retrieved.

[*symbol*] TOPTION [EP = *endpoint_id*]
[,OPLEN = *protocol_options_length*]
[,OPBUF = *protocol_options_address*]
[,OPALET = *protocol_options_alet*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,NEGOT | NONEGOT]
[,DECLARE | VERIFY | QUERY | DEFAULT]
[,TP | API])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TOPTION macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

OPLEN = *protocol_options_length*

Indicates the length (in bytes) of the protocol option list identified by the OPBUF operand. A value of zero indicates there is no protocol option list, and is invalid for the TOPTION macro instruction.

Default: 0 (no protocol option list)

OPBUF = *protocol_options_address*

Indicates the address of a storage area containing a protocol option list. The area must contain a list of variable-length protocol options, with each option identified by its length and name. Each entry in the list must also contain room for an option value, which is initialized with the desired value of the option for the DECLARE and VERIFY forms of the TOPTION macro instruction. For the DEFAULT and QUERY forms, the option value is returned in the storage area provided.

The type, number and format of protocol options is provider-dependent, and the maximum size of the option list can be determined by issuing a TINFO macro instruction. The storage area can be aligned on any boundary convenient to the application program.

Default: 0 (no protocol option list)

OPALET = *protocol_options_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the OPBUF parameter. The OPALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The OPALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TOPTION macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TOPTION request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = NEGOT | NONEGOT

Indicates whether protocol options associated with this request can be negotiated to an inferior value.

If the option is OPTCD=NEGOT, protocol options are negotiated to comply with the limits of the transport provider, and the conditional completion code is set to indicate that the negotiation occurred.

If the option is OPTCD=NONEGOT, negotiation is disallowed, and unacceptable options result in the abnormal completion of the TOPTION macro instruction.

Default: NONEGOT (negotiation disallowed)

OPTCD = DECLARE | VERIFY | QUERY | DEFAULT

Indicates an action to be performed by the TOPTION macro instruction. Protocol options that are the subject of the actions listed in this table are contained or returned in an option list designated by the OPLEN and OPBUF operands. One of these actions may be indicated:

If the option is OPTCD=DECLARE, the options specified by the application program are invoked, and the option list is updated with the inferior value of any negotiated options.

If the option is OPTCD=VERIFY, the options specified by the application program are verified, and the option list is updated with the inferior value of any negotiated options.

If the option is OPTCD=QUERY, the current value of options selected by the application program are returned.

If the option is OPTCD=DEFAULT, the default value of options selected by the application program are returned.

The type, number and format of protocol options supported by a transport provider is protocol specific.

Default: DECLARE (invoke protocol options)

OPTCD = TP | API

Indicates whether the option list identified by the OPBUF and OPLEN operands contains transport interface or transport provider options.

If the option is OPTCD=API, the option list contains interface options that are processed solely by the API.

If the option is OPTCD=TP, the option list is passed to the transport provider for processing.

Transport interface and transport provider options can only be manipulated with separate invocations of the TOPTION macro instruction.

Default: TP (transport provider options)

ECB = INTERNAL | event_control_block_addr

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TOPTION macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = tpl_exit_routine_address	Indicates the address of an exit routine to be scheduled when the TOPTION macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB. This operand is mutually exclusive with the previous ECB operand. Default: not indicated (no TPL exit routine)
MF = (I | L | G | M | E, [tpl_address])	Indicates the standard, list, generate, modify, or execute form of the TOPTION macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used. Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters. Default: MF=I (standard, nonreentrant form)

Completion Information

The TOPTION macro instruction completes normally (or conditionally) when the protocol options specified by the application program have been processed by the transport provider. If OPTCD=DECLARE was specified, the indicated options have been negotiated and set to the requested values. Negotiated, verified, or queried values of the indicated options are returned in the storage area provided by the application program. The state of the endpoint is not changed.

On return to the application program, the general return code in register 15 is set to 0 (TROKAY), and a conditional completion code is returned in register 0. TCNEGOT is set in the conditional completion code if any of the protocol options specified by the application program were negotiated to an inferior value, and OPTCD=DECLARE was indicated. TCVERIFY is set if any of the options specified by the application program are not supported by the transport provider, and OPTCD=VERIFY was indicated. The TPL return code field is set accordingly. No other information is returned.

If the TOPTION macro instruction completes abnormally, no protocol options are negotiated, set, or returned. All options in effect at the time the TOPTION macro instruction was executed remain in effect. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TOPTION return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-33 TOPTION Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/ Explanation		
TROKAY	TAOKAY	TCNEGOT	TCVERIFY	
TRFAILED	TAEXCPTN	TENONEGO		
	TAENVIRO	TESYSERR TESTOP TEUNAUTH	TESUBSYS TETERM TERSOURC	TEDRAIN TEUNSUPO
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD TEBDOPTN	TEBDECB
	TAPROCED	TEAMODE	TESTATE	TEINCMPL
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TOPTION macro instruction is used to manage protocol options associated with an endpoint. Protocol options can be specified by the application program, and options the application program intends to specify can first be verified to determine if they are supported by the transport provider. Default options that are in effect if not overridden by the application program can be retrieved, and the current value of options in effect for the endpoint can be queried. The action to be taken by the TOPTION macro instruction is indicated by the OPTCD operand.

Multiple options can be manipulated with a single invocation of the TOPTION macro instruction. The options are identified by an option list provided by the application program, and updated by the transport provider. Each entry in the list represents one option, and contains the length and name of the option. The format of the option name is provider-dependent, but usually identifies a protocol level and option supported by the protocol. Each entry also contains room for an option value, whose length is option-dependent. The format of an option list entry for a specific transport provider is defined by the TPO dsect (see TDSECT).

This table illustrates the general format:

Figure 1-1 TOPTION Format of Option Name

x+0	OPTION LENGTH	OPTION NAME
x+4	OPTION VALUE	
x+optlen		

The type, number, and format of protocol options supported by the transport provider is protocol and provider dependent. The application should be conservative in its use of protocol options to remain independent of a specific transport provider. The options supported by specific transport providers, and the format of their specification, are documented in the provider-specific appendix at the end of this reference.

Although a transport provider supports a particular option, there is no guarantee it can support the value requested at the time of the request. If necessary, the transport provider negotiates the option to an inferior value in order to successfully complete the macro instruction.

- If OPTCD=NONEGOT is specified, no negotiation is allowed, and the macro instruction is completed abnormally.
- If OPTCD=NONEGOT is not specified, the macro instruction is completed conditionally, and the fact that an option was negotiated is indicated by the conditional completion code. The option list is updated with the negotiated option value.

Transport Provider Options

The TOPTION macro instruction is generally used to manipulate transport provider options. However, if so indicated by the OPTCD operand, the TOPTION macro instruction can also be used to manipulate transport interface options. Transport interface (OPTCD=API) and transport provider (OPTCD=TP) options cannot be combined in the same options list. The API option names are defined by the TPO dsect. These names correspond to the symbols used in the dsect to define the option name.

These names describe OPTCD=API options. The values are all four (4) bytes long so the option length must be eight (8) bytes.

Table 1-34 describes TOPTION OPTCD=API Options

Table 1-34 TOPTION OPTCD=API Options

TPOAQSND	The maximum number of TSEND or TSENDTO macro instructions that can be executed at an endpoint without waiting for at least one to complete. In other words, TPOAQSND is the maximum number of pending send requests.
TPOAQRCV	The maximum number of TRECV or TRECVFR macro instructions that can be executed at an endpoint without waiting for at least one to complete. In other words, TPOAQRCV is the maximum number of pending receive requests.
TPOALSND	The maximum number of bytes of data that can be pending for outstanding TSEND or TSENDTO requests. In other words, TPOALSND is the total amount of send buffering allocated for an endpoint.
TPOALRCV	The maximum number of bytes of data that can be pending for outstanding TRECV or TRECVFR requests. In other words, TPOALRCV is the total amount of receive buffering allocated for an endpoint.

TCP Provider Session Options

These options are valid only for TCP provider sessions. These names describe OPTCD=TP options. The values are all four (4) bytes long, so the option length must be eight (8).

Table 1-35 describes TOPTION TCP Provider Option

Table 1-35 TOPTION TCP Provider Options

TPOPRWND	The size of the receive buffer used by TCP. This is reflected as the receive window advertised by TCP. This option is valid only for TCP sessions and must be set before a connection is established. The range of acceptable values is 256-500,000. Values above 65535 are valid, and allocates a buffer of the specified size. A sliding window of 65535 is used in this buffer by TCP. Default is 261376.
TPOPKTIM	The interval of idle time used by the keepalive option specified in minutes. The range of acceptable values is 1-1439. The default is 120.
TPOPKEEP	The keepalive option enables the periodic probing of the remote. This option specifies the type of keepalive to use. It is also used to turn off the keepalive option. These integer values are supported: 0 – Turn off keepalive 1 – Use keepalive with no data, and do not abort the session if no response 2 – Use keepalive with no data, and abort the session if no response. 3 – Use keepalive with data See the notes following this table for a discussion of keepalive. Their use is discouraged and is provided only for those applications that are incapable of detecting idle sessions.
TPOPDNAG	Defeat TCP's Nagle algorithm used to gather send data into maximal packets. A value of one defeats the Nagle algorithm. A value of zero restores normal operation of the Nagle algorithm. This option is only valid when a connection is established. It is intended for use by applications that send small amounts of data and for performance reasons demand that they be sent in small packets. High volume applications should not use this option and should use multiple asynchronous TSENDs instead.

Note These notes apply to the keepalive option (in other words, TPOPKEEP):

The use of keepalive is discouraged by the internet community. It should be the responsibility of the application to detect idle connections and probe them or terminate them as appropriate.

However, here is a brief description of their use:

- The keepalive options may only be set when a connection is established. Once set, they do not become effective until there is network activity, either a TSEND or a TRECV. This also applies to turning keepalive off.
- Keepalive packets may be sent with or without data. Normally they are sent without data. Since some implementations do not respond to keepalive of this form, excessive retransmissions of the keepalive does not abort the session. However, if the session has terminated at the remote end, that host sends a reset, aborting the connection. If the host does not respond at all, you can request that the session be aborted after excessive retransmissions.
- If it is determined that the remote host implementation does not respond to a keepalive with no data, you can request that keepalive be sent with one byte of data. The retransmission mechanism aborts the session if retransmissions are exceeded.

See RFC 1122 for a more complete description of keepalive considerations.

TCP/UDP/RAW Provider Session Options

These options are derived from the Berkeley socket implementation and are valid for TCP, UDP or RAW sessions. These options are used with OPTCD=TP options.

Table 1-36 TOPTION TCP/UDP/RAW Provider Session Options

TPOIPOPT

Set or get options for IP protocol

The maximum total length of IP options is 40 bytes.

Most IP protocol options are itemized in standard IP header format starting with:

- Option Code: 1 byte
- Length of segment: 1 byte
- Pointer to first variable data field: 1 byte
- Variable length data fields

Supported options are:

- 0 End-of-options (This option is a single byte and does not use the standard IP option format.)
- 1 No-op (This option is a single byte and does not use the standard IP option format.)
- 7 Record Route Option

Code	Length	Pointer	4-byte IP Address1	.....	4-byte IP Address9
7	7-39	4			

- 68 Timestamp Option

  The Timestamp Option uses two different formats depending on the operation requested within the Timestamp Option Flags field.

Code	Length	Pointer	Overflow	Flags	4-byte IP Address1	4-byte Timestamp #1	.....	4-byte IP Address4	4-byte Timestamp #4
68	12-36	5							

Code	Length	Pointer	Overflow	Flags	4-byte Timestamp #1	.....	4-byte Timestamp #9
68	8-40	5					

  Note: Overflow and Flags fields are 4 bits each.

  Overflow: count of additional hops not timestamped

  Flags:

  — 0 = Record timestamps only

  — 1 = Record IP address/timestamp pairs

  — 2 = Record timestamps for pre-set IP addresses

- 131 Loose Source and Record Route Option

Code	Length	Pointer	4-byte IP Address1	.....	4-byte IP Address9	4-byte IP Destination Address
131	11-43	4				

  Route through the specified addresses; additional hops may be taken. The last address in the list must be the final destination.

- 137 Strict Source and Record Route Option

Code	Length	Pointer	4-byte IP Address1	.....	4-byte IP Address9	4-byte IP Destination Address
137	11-43	4				

Route only through each address as specified. The last address in the list must be the final destination.

Table 1-36 TOPTION TCP/UDP/RAW Provider Session Options (Continued)

TPOSIOAR	Add a single routing table entry. This socket option is for internal use only. The option is failed when requested from outside the transport provider address space. This option requires an 88 byte field to specify the route data. The route data is mapped by T01DIRT (internal DSECT macro).
TPOSIODR	Delete a single routing table entry. This socket option is for internal use only. The option is failed when requested from outside the transport provider address space. This option requires an 88 byte field to specify the route data. The route data is mapped by T01DIRT (internal DSECT macro).
TPOIFNO	Get the number of interfaces. Provide a 4-byte buffer for return of the count.
TPOSIFCF	Get the interface configuration list. The list is returned in a buffer which is sized by (number of interfaces * 32) + 4, where the first four bytes contain the length of the configuration list. Fields returned for each interface are: Offset Length Description 0 16 Name of the interface 16 16 Interface network address The first 8 bytes of the interface network address can be mapped using the Transport Protocol Address (TPA) DSECT. The last 8 bytes are padding.
TPOSIFLG	Get interface flags. Provide an 18-byte buffer with the interface name specified in the first 16 bytes. Associated flags are returned in bytes 17-18. See data structure TIOC for flag definitions.
TPOSIFMT	Get maximum transmission unit. Provide a 20-byte buffer with the interface name specified in the first 16 bytes. The associated MTU is returned in bytes 17-20.
TPOSIFME	Get metric. Provide a 20-byte buffer with the interface name specified in the first 16 bytes. The associated metric is returned in bytes 17-20.
TPOSIFNM	Get network address mask. Provide a 32-byte buffer with the interface name specified in the first 16 bytes. The associated network mask is returned in bytes 17-24. This information can be mapped using the Transport Protocol Address (TPA) DSECT (for example, the address mask is at offset +4 within this structure). Bytes 25-32 are padding.
TPOSIFBA	Get the broadcast address. Provide a 32-byte buffer with the interface name specified in the first 16 bytes. The associated address is returned in bytes 17-24. This information can be mapped using the Transport Protocol Address (TPA) DSECT (for example, the broadcast address is at offset +4 within this structure). Bytes 25-32 are padding.
TPOSIFAD	Get the interface address. Provide a 32-byte buffer with the interface name specified in the first 16 bytes. The associated address is returned in bytes 17-24. This information can be mapped using the Transport Protocol Address (TPA) DSECT (for example, the interface address is at offset +4 within this structure). Bytes 25-32 are padding.
TPOSIFEN	Get the hardware address. Provide a 32-byte buffer with the interface name specified in the first 16 bytes. The associated address is returned beginning at byte 17 and extending for a length that is dependent on the link layer addressing that is in use (in other words, the length would be 6 in the case of Ethernet). The remainder of the buffer is padding.
TPOSIFDS	Get the destination address. Provide a 32-byte buffer with the interface name specified in the first 16 bytes. The associated address is returned in bytes 17-24. This information can be mapped using the Transport Protocol Address (TPA) DSECT (for example, the destination address is at offset +4 within this structure). Bytes 25-32 are padding.
TPOIPTTL	Set or get IP time-to-live. Provide a 4-byte field for the maximum number of routing hops to be taken in a range from 1 to 255.
TPOIPTOS	Set or get type of service. Provide a 4-byte field for the type of service in a range from 0 to 255.
TPOTPMSS	Set or get maximum segment size. Provide a 4-byte field for the maximum segment size in a range from 512 to 64K-20. The number may reflect the maximum transmission unit size less the size of the IP header (20 bytes).
TPOIPDNR	Set or get IP do-not-route. Provide a 4-byte field. A value of one indicates that data not be sent through any router, and is restricted to destinations on the local network.

Table 1-36 TOPTION TCP/UDP/RAW Provider Session Options (Continued)

TPOIPBRO	Set or get IP broadcast. Provide a 4-byte field. A value of one indicates that broadcasting is allowed.
TPOUDSUM	Set or get UDP checksums option. Provide a 4-byte field. A value of one indicates that UDP checksumming is in effect.
TPOREUSE	Set or get reuse address option. Provide a 4-byte field. A value of one indicates for TBIND to use a server port number even though the port number is in use by another server.
TPOUDATA	Set user data. Provide a 24-byte field with character data to be displayed by NETSTAT. The 24 byte field breaks down as follows: Bytes — Value 0-7 — User ID 8-15 — Secondary Logical Unit (SLU) 16-23 — Primary Logical Unit (PLU) or service Cisco IOS for S/390 does not verify the contents of the data, it simply accepts it and assumes that the TU invoking TOPTION is correct.

The TOPTION macro instruction can be issued when the endpoint is in the opened (TSOPENED), disabled (TSDSABLD), enabled (TSENABLD), and connected (TSCONNCT) states. However, for COTS endpoints, some protocol options may not be changed after the endpoint is connected. In the future, some protocol options may also be specified with the TCONNECT and TACCEPT macro instructions. Similarly, in the future, for CLTS endpoints operating in pure datagram mode (in other words, without associations), options may be specified with each datagram sent.

TPL

Create a Transport Service Parameter List

The TPL macro instruction is used to create a Transport Service Parameter List (TPL) which is the primary argument of all transport service functions.

[*symbol*] TPL [EP = *endpoint_id*]
[,ADLEN = *protocol_address_length*]
[,ADBUF = *protocol_address_address*]
[,ADALET = *protocol_address_alet*]
[,DALEN = *user_data_length*]
[,DABUF = *user_data_address*]
[,DAALET = *user_data_alet*]
[,OPLEN = *protocol_options_length*]
[,OPBUF = *protocol_options_address*]
[,OPALET = *protocol_options_alet*]
[,QLSTN = *listen_queue_length*]
[,NEWEP = *new_endpoint_id*]
[,SEQNO = *sequence_number*]
[,USER = *endpoint_userid*]
[,TCB = *task_control_block_address*]
[,ASCB = *address_space_control_blk_addr*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,TRUNC | NOTRUNC]
[,NEGOT | NONEGOT]
[,BLOCK | NOBLOCK]
[,ASSIGN | USE]
[,LOCAL | REMOTE]
[,PRIMARY | SECNDRY | STATS]
[,DECLARE | VERIFY | QUERY | DEFAULT]
[,TP | API]
[,MORE | NOMORE]
[,NORMAL | EXPEDITE]
[,EOM | NOTEOM]
[,DIRECT | INDIR]
[,ABORT | CLEAR]
[,DELETE | PASS]
[,TUB | ACEE]
[,PLAIN | CIPHER])]
[,NOFULL | FULL]
[,NOTIMEOUT | TIMEOUT]
[,MBUF | NOMBUF])]
[,FNCCD = TACCEPT | TADDR | TBIND | TCLEAR |
TCLOSE | TCONFIRM | TCONNECT |
TDISCONN | TINFO | TLISTEN | TOPTION |
TRECV | TRECVERR | TRECVFR |
TREJECT | TRELACK | TRELEASE | TRETRACT |
TSEND | TSENDTO | TUNBIND | TUSER]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (L | M, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint associated with the TPL. When the TPL is later executed using a TEXEC or other API macro instruction, the requested function is performed on the indicated endpoint. The value specified must be a valid endpoint ID returned from a TOPEN macro instruction.

Default: 0 (no endpoint specified)

ADLEN = *protocol_address_length*

Indicates the length (in bytes) of the protocol address storage area identified by the ADBUF operand. If the storage area contains a protocol address to be supplied to the transport provider, the length indicated should be the actual length of the protocol address. If the storage area is to be used by the transport provider for returning a protocol address, the length indicated should be the maximum length of the storage area. The transport provider updates the ADLEN field in the TPL to indicate how many bytes were returned.

A length of zero may be specified indicating there is no protocol address in the storage area, or one is not to be returned by the transport provider, depending on the semantics of the request. If the length is non-zero, ADBUF must be coded and indicate a valid storage area.

Default: 0 (no protocol address)

ADBUF = *protocol_address_address*

Indicates the address of a protocol address storage area whose length is specified by the ADLEN operand.

If the semantics of the request require a protocol address to be supplied to the transport provider. the storage area must contain a valid protocol address.

If the semantics of the request require the transport provider to return a protocol address, the storage area is updated, and should be large enough to contain the protocol address. The maximum size of a protocol address may be obtained from the transport provider with the TINFO macro instruction.

An address of zero may be specified indicating there is no protocol address to be supplied or returned. However, if the address is zero, the length must also be zero. There are no alignment restrictions for this storage area. The length of a protocol address storage area is variable to accommodate a variety of transport providers, or to accommodate transport protocols that use variable length protocol addresses. The structure and content of a protocol address is provider-dependent. Information for specific transport providers can be found in the appropriate appendix at the end of this manual.

Default: 0 (no protocol address storage area)

ADALET = *protocol_address_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the ADBUF parameter. The ADALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The ADALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

DALEN = *user_data_length* Indicates the length (in bytes) of the data storage area identified by the DABUF operand.

If the storage area contains data to be supplied to the transport provider (for example, a send request), the length indicates that it should be the actual length of the data.

If the storage area is to be used by the transport provider for returning data (for example, a receive request), the length indicates that it should be the maximum length of the storage area. The transport provider updates the DALEN field in the TPL to indicate how many bytes were returned.

A length of zero may be specified indicating there is no data in the storage area, or none is to be returned by the transport provider, depending on the semantics of the request. If the length is non-zero, DABUF must be coded and indicate a valid storage area. Specifying a length of zero may be invalid for certain types of requests, and if so, generates an error.

Default: 0 (no user data)

DABUF = *user_data_address* Indicates the address of a data storage area whose length is specified by the DALEN operand.

If the semantics of the request require data to be supplied to the transport provider, the storage area must contain the required user data.

If the semantics of the request require the transport provider to return data, the storage area is updated, and should be large enough to contain the desired amount of data.

An address of zero may be specified indicating there is no data to be supplied or returned. However, if the address is zero, the length must also be zero. There are no alignment restrictions for this storage area.

The contents of the storage area varies depending on the type of request (for example, if the function requested is TCONNECT, the data storage area contains connect user data). Similarly, if the function is TDISCONN, the data storage area contains disconnect user data. For TRECV and TSEND functions, the storage area contains arbitrary application data. A given transport provider may not support all data types. The API does not interpret the content of any data contained in the storage area.

Just as the content of the data storage area varies depending on the transport function requested, so does the maximum length. The maximum size of various data types may be obtained from the transport provider with the TINFO macro instruction.

Default: 0 (no user data storage area)

DAALET = *user_data_alet* Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the DABUF parameter. The DAALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The DAALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

OPLEN =
protocol_options_length

Indicates the length (in bytes) of the protocol options storage area identified by the OPBUF operand.

If the storage area contains protocol options to be supplied to the transport provider, the length indicates that it should be the actual length of the protocol options.

If the storage area is to be used by the transport provider for returning protocol options, the length indicates that it should be the maximum length of the storage area. The transport provider updates the OPLEN field in the TPL to indicate how many bytes were returned.

A length of zero may be specified indicating there are no protocol options in the storage area, or none are to be returned by the transport provider, depending on the semantics of the request. If the length is non-zero, OPBUF must be coded, and indicate a valid storage area.

Default: 0 (no protocol options)

OPBUF =
protocol_options_address

Indicates the address of a protocol options storage area whose length is specified by the OPLEN operand.

If the semantics of the request require protocol options to be supplied to the transport provider, the storage area must contain valid protocol options.

If the semantics of the request require the transport provider to return protocol options, the storage area Is updated, and should be large enough to contain the protocol options. The maximum size of protocol options may be obtained from the transport provider with the TINFO macro instruction.

An address of zero may be specified indicating there are no protocol options to be supplied or returned. However, if the address is zero, the length must also be zero. There are no alignment restrictions for this storage area.

The length of a protocol options storage area is variable to accommodate a variety of transport providers, or to accommodate transport protocols that use variety of protocol options. The type, number and format of protocol options is provider-dependent. Information for specific transport providers can be found in the appropriate appendix at the end of this manual.

Default: 0 (no protocol options storage area)

OPALET =
protocol_options_alet

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the OPBUF parameter. The OPALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The OPALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

QLSTN = *listen_queue_length*

Indicates the size of the queue for holding incoming connections arriving at an endpoint, and pending connect indications received by the application program.

If the value specified is zero, no connections can be queued, and the endpoint is disabled for receiving connections.

If the value specified is non-zero, incoming connections are queued, and corresponding connect indications are generated at the endpoint.

A connect indication remains pending until it is accepted or rejected by the application program, or until the connection is abandoned by the caller. The value of this operand generally determines whether the application program is operating in client or server mode.

The transport provider may not be able to queue the number of connections specified by the application program, and as a result, attempts to negotiate the indicated value to a lesser amount. If negotiation is permitted by the application program (OPTCD=NEGOT), the request completes conditionally, and returns a conditional completion code in register 0. Otherwise, the TBIND request completes abnormally.

Default: 0 (endpoint is disabled)

NEWEP = *new_endpoint_id*

Specifies the endpoint at which a connection is established when a TACCEPT function is executed at an endpoint with a pending connect indication. The SEQNO operand specifies which connect indication is being accepted. The value specified for NEWEP is the endpoint ID returned from a TOPEN macro instruction. A value of zero may be used to indicate the listening endpoint.

The connected endpoint may be the endpoint at which the connect indication arrived, or a new endpoint.

If NEWP specifies a new endpoint, the endpoint must be in the disabled (TSDSABLD) state. A new endpoint must have a local protocol address bound to it before a connect indication can be accepted. The local protocol address may be the same as that bound to the listening endpoint, or different (if supported by the underlying protocol).

If NEWP specifies an existing endpoint, NEWP is the endpoint ID of the listening endpoint (or 0). The endpoint must not have any pending connect indications other than the one being accepted (in other words, the endpoint must be in the connect-indicating-pending state (TSINCONN), and the number of queued indications must be one).

Default: 0 (connection established at listening endpoint)

SEQNO = *sequence_number*

Used by the TACCEPT or TREJECT macro instructions to specify which of potentially several pending connect indications is to be accepted or rejected. The specified value is the sequence_number that was returned by the transport provider with the completion of a TLISTEN macro instruction, previously executed at the same endpoint.

Default: 0 (most likely an invalid sequence number)

USER = *endpoint_userid*

Associates a user ID with the endpoint for authorization and accounting purposes.

If the option is OPTCD=TUB, the specified value must be the address of a Transport Endpoint User Block (TUB) containing the user information.

If the option is OPTCD=ACEE, the specified value must be the address of an Accessor Environment Element (ACEE) obtained from the local security system when the user ID was authenticated.

If the option is not coded, the application name specified in the APCB is used.

The password contained in the TUB may be plain text or cipher text depending on the OPTCD=PLAIN | CIPHER operand. If cipher text, it is assumed that the password was encrypted using the encryption mechanism supplied by the local security system. The API merely provides the password to the security system in its encrypted form.

The user ID or application name is also supplied to the transport provider. How this information is used is unspecified, and provider-dependent.

Default: 0 (no user ID; use application name for accounting and authorization)

TCB = *task_control_block_address*

Used by the TCLOSE macro instruction to specify the TCB address of a task that is to receive control of an endpoint when closed with OPTCD=PASS. A value of zero causes the endpoint to be passed to any task that issues a complementary TOPEN macro instruction.

Default: 0 (pass to any task)

ASCB = *address_space_control_blk_addr*

Used by the TCLOSE macro instruction to specify the ASCB address of an address space that is to receive control of an endpoint when closed with OPTCD=PASS. A value of zero is taken to mean the current address space.

Default: 0 (pass to current address space)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

When a request is issued in asynchronous mode, control is immediately returned to the application program after the API has accepted the request. When the requested function has completed, the API does one of these actions:

If an ECB has been specified, (either internal or external), the API posts a completion indicator in the event control block. The application program must issue a TCHECK or system WAIT macro instruction to determine whether the ECB has been posted. If a system WAIT or similar technique is used, the application program must still issue a TCHECK macro instruction to mark the TPL inactive, and to cause entry to the SYNAD or LERAD exit routine if the request completes with an error.

If the EXIT operand is in effect for the TPL, the API schedules the exit routine indicated by this operand. The TPL exit routine should issue the TCHECK macro instruction to set the TPL inactive, and to cause entry into a SYNAD or LERAD exit routine if the requested function completed with an error. TCHECK must be issued even if a SYNAD or LERAD exit routine have not been provided.

Note For more information on task synchronization, read Cisco IOS for S/390 Assembler API Concepts.

The application program should not modify an active TPL. A TPL is considered active from the time a request has been accepted until it is marked inactive by the TCHECK macro instruction. Modifying an active TPL yields unpredictable results, and may cause the request or application program to terminate abnormally.

The coding of the ECB and EXIT operands must be consistent with the OPTCD operand. For example, if OPTCD=SYNC is indicated, the ECB and EXIT operands should not be coded. However, since synchronous mode uses an internal ECB, ECB=INTERNAL is permitted when OPTCD=SYNC is indicated. Also, if an external ECB or exit routine is specified, the default synchronization mode is changed to asynchronous. If OPTCD=SYNC is indicated in this case, an error message is generated at assembly time.

Default: SYNC (synchronous mode)

OPTCD = TRUNC | NOTRUNC

Indicates whether items being returned to the application program by the transport provider are to be truncated if they do not fit within the storage area provided. These actions are taken in accordance with the setting of this option:

If the option is OPTCD=TRUNC, the data or information being returned is truncated at the end of the storage area, and the request is completed conditionally as long as no other errors occur. The length of the storage area as provided by the application program remains unchanged, and the residual data is discarded. TCTRUNC is set in the conditional completion code.

If the option is OPTCD=NOTRUNC, no data or information is written into the storage area, and the request is completed abnormally. The storage area length is not changed, and there is no way for the application program to determine how much of a deficit there was.

If OPTCD=TRUNC is indicated, and a transport function completes conditionally with TCTRUNC set, some significant data may have been lost. The application programmer should be cautious when invoking this option, and should be fully aware of what data or information might be discarded. The TINFO macro instruction may be used to determine the maximum lengths for various data types supported by the transport provider.

Default: NOTRUNC (truncation disallowed)

OPTCD = NEGOT | NONEGOT

Indicates whether protocol options associated with a request can be negotiated to an inferior value if the transport provider cannot support the option as specified. These actions are taken in accordance with the setting of this option:

If the option is OPTCD=NEGOT, the transport provider is free to negotiate an option to an inferior value in order to complete the requested function. If no other errors occur, the request is completed conditionally. TCNEGOT is set in the conditional completion code. In some cases, the negotiated value is returned to the application program at the completion of the request, and in others, an additional macro instruction must be issued to retrieve the negotiated value.

If the option is OPTCD=NONEGOT, the transport provider may not negotiate an option to an inferior value. If the transport provider cannot support the option as indicated, the request is completed abnormally.

OPTCD=NEGOT can be used to provide a degree of independence from the transport provider and the particular protocol in use.

If the application has requested a certain quality of service, a lower quality of service may work nearly as well, albeit with diminished performance.

Portability of an application is enhanced when the dependence on protocol options is minimized.

Default: NONEGOT (negotiation disallowed)

OPTCD = BLOCK | NOBLOCK

Indicates whether or not the issuing task can be suspended if the TPL macro instruction cannot be completed immediately.

If the option is OPTCD=BLOCK (and no connect indicated has been generated), the issuing task is suspended until a connection request arrives.

If the option is OPTCD=NOBLOCK, the macro instruction is completed immediately, and an abnormal return code indicates that the task would have been suspended for an indefinite period of time.

The TPL macro instruction can be used to poll for new connect indications. If a connect indication is available, the request is completed as usual. Otherwise, the request is completed abnormally and the transport user can try again after delaying an appropriate period of time.

In either case, if a connect indication has already been generated, the TPL macro instruction completes normally without suspending the issuing task.

Default: BLOCK (suspend issuing task if necessary)

OPTCD = ASSIGN | USE

Indicates in a TBIND macro instruction whether a protocol address provided by the application program is to be bound to the endpoint, or the transport provider is to assign a local protocol address and return the value to the application program.

If the option is OPTCD=ASSIGN, if OPTCD=ASSIGN is indicated, the transport provider assigns an address and return the value in the storage area designated by the ADLEN and ADBUF operands.

If the option is OPTCD=USE, then if OPTCD=USE is indicated, the ADLEN and ADBUF operands designate a storage area that must contain a valid protocol address.

Default: USE (use protocol address provided)

OPTCD = LOCAL | REMOTE

Indicates in a TADDR macro instruction whether the transport provider should return the local protocol address bound to the indicated endpoint, or the remote protocol address connected to, or associated with, the endpoint.

If the option is OPTCD=LOCAL, a local address must have already been bound to the endpoint.

If the option is OPTCD=REMOTE, the endpoint must be connected to (for connection-mode service), or associated with (for connectionless-mode service), a remote protocol address.

If either of these conditions are violated, the TADDR function is completed abnormally.

Default: LOCAL (return local protocol address)

OPTCD = PRIMARY | SECNDRY | STATS

Indicates the type of information requested with a TINFO macro instruction. The information type must one of these options:

PRIMARY - Designates primary protocol information whose format and meaning is standardized for all transport providers. The application program can use this information to determine the basic characteristics of the transport service and limits of the transport provider.

SECNDRY - Designates secondary protocol information whose format and meaning is specific to the transport service being used. This information includes internal protocol and state variables that govern the operation of the transport protocol. Transport providers are not required to support this option code. Refer to the appropriate appendix at the end of this reference for more information concerning a specific transport provider.

STATS - Designates statistical information recorded by the transport provider whose format and meaning is specific to the transport service being used. Transport providers are not required to support this option code. Refer to the appropriate appendix at the end of this reference for more information concerning a specific transport provider.

Default: PRIMARY (return basic protocol information)

OPTCD = DECLARE | VERIFY | QUERY | DEFAULT

Indicates an action to be performed by the TOPTION macro instruction. Protocol options that are the subject of these actions are contained or returned in an option list designated by the OPLEN and OPBUF operands. One of these actions may be indicated:

DECLARE - The options specified by the application program are invoked, and the option list is updated with the inferior value of any negotiated options.

VERIFY - The options specified by the application program are verified, and the option list is updated with the inferior value of any negotiated options.

QUERY - The current value of options selected by the application program are returned.

DEFAULT - The default value of options selected by the application program are returned.

The type, number, and format of protocol options supported by a transport provider are protocol specific. Refer to the appropriate appendix at the end of this reference for more information on specific transport providers.

Default: DECLARE (invoke protocol options)

OPTCD = TP | API

Indicates whether the option list identified by the OPBUF and OPLEN operands contains transport interface or transport provider options.

If the option is OPTCD=API, the option list contains interface options that are processed solely by the API.

If the option is OPTCD=TP, the option list is passed to the transport provider for processing.

Transport interface and transport provider options can only be manipulated with separate invocations of the TOPTION macro instruction.

Default: TP (transport provider options)

OPTCD = MORE | NOMORE

Indicates for a TSEND macro instruction whether the application program intends to immediately send more data, or intends to pause momentarily until it has more data to send.

If the option is OPTCD=MORE, the application program expects to immediately issue another TSEND macro instruction at the same endpoint.

If the option is OPTCD=NOMORE, the application program has no more data to send, but intends to leave the connection established, and may resume sending data later.

The interpretation of this option code by the transport provider is protocol dependent. The intent is that the transport provider uses this information to augment its packetizing algorithm, and deduce when unsent data must be forwarded on the connection. Not all connection-mode transport providers are required to interpret this option code, but all are required to accept it. If the TSEND macro instruction is executed in synchronous mode, OPTCD=MORE is ignored.

No implication is drawn about message boundaries by the indication of OPTCD=NOMORE. If the underlying transport protocol can preserve logical boundaries within the data stream, then such boundaries should be indicated with OPTCD=EOM.

Default: NOMORE (send data immediately)

OPTCD = NORMAL | EXPEDITE

Indicates for a TSEND macro instruction whether the user data should be sent as normal or expedited data.

If the option is OPTCD=NORMAL, the data associated with the request is to be sent as normal data.

If the option is OPTCD=EXPEDITE, the data is to be sent as expedited data.

The distinction between normal and expedited data is left to the interpretation of the transport provider.

Default: NORMAL (send data as normal data)

OPTCD = EOM | NOTEOM

Indicates whether the data associated with a TSEND or TSENDTO request is a complete message or datagram, or is continued with one or more subsequent macro instructions.

If the option is OPTCD=EOM, the last byte of data corresponds to the end of the message or datagram.

If the option is OPTCD=NOTEOM, the end of the message or datagram does not occur with this request, and is continued with at least one more TSEND or TSENDTO macro instruction.

Transport providers operating in connectionless-mode are not required to accept it. If supported, the notion of a message (or TSDU) is synonymous with datagram, and the significance of the OPTCD=NOTEOM indication is only a local phenomenon.

Default: EOM (end of TSDU or datagram)

OPTCD = DIRECT | INDIR

Indicates the format of the user data parameter.

If the option is OPTCD=DIRECT, the DABUF and DALEN operands identify a storage area into which data should be received directly.

If the option is OPTCD=INDIR, the storage area identified by these operands contains an indirect data vector. An indirect data vector consists of a list of address-length pairs, with each element identifying a separate segment of non-contiguous storage. In this case, DABUF is the address of the first element in the list, and DALEN is the total length of the list. The length of the vector must be a multiple of eight, and the total amount of data that can be received is the sum of the lengths of each data segment.

Default: DIRECT (send directly from data area)

OPTCD = ABORT | CLEAR

Indicates the action to be taken by the transport provider when the application program issues a TDISCONN, TREJECT, or TRELACK macro instruction at an endpoint for which a disconnect indication is already pending.

If the option is OPTCD=ABORT, the request is completed abnormally, and the application program must clear the pending disconnect indication with a TCLEAR macro instruction.

If the option is OPTCD=CLEAR, the TDISCONN, TREJECT, or TRELACK macro instruction is completed normally (or conditionally) if no other errors occur.

Default: CLEAR (clear pending disconnect indication)

OPTCD = DELETE | PASS

Indicates the disposition of the endpoint designated in the TPL associated with a TCLOSE macro instruction.

If the option is OPTCD=DELETE, the endpoint is closed, and any record of the endpoint is deleted from all internal tables and local storage.

If the option is OPTCD=PASS, the endpoint is not closed, and control of the endpoint is passed to the designated task or address space.

When control is being passed, the TCLOSE request does not complete until a complementary TOPEN (OPTCD=OLD) macro instruction has been issued by the acquiring task or address space.

Default: DELETE (delete endpoint)

OPTCD = TUB | ACEE

Indicates the format of user ID information referenced by the USER operand.

If the option is OPTCD=TUB, user ID, group, and password information are provided in a Transport User Block (TUB).

If the option is OPTCD=ACEE, the user information is contained in an Accessor Environment Element (ACEE) obtained from the local security system.

Default: TUB (user information provided in TUB)

OPTCD = PLAIN | CIPHER

Indicates whether the password contained in the Transport User Block (TUB) designated with the USER operand has been encrypted, or is in its plain text form.

If the option is OPTCD=PLAIN, the password is in plain text.

If the option is OPTCD=CIPHER, the password is encrypted.

The API uses this information when requesting user ID and password verification from the local security system.

Default: PLAIN (password in plain text)

OPTCD = NOFULL | FULL

Indicates completion processing for this request based on the amount of data.

If the option is OPTCD=NOFULL, this request is completed as soon as any data arrives.

An option of OPTCD=FULL indicates that this request is not to be completed until either a specified timeout occurs, or the requested amount of data arrives to fill up this request.

Note: Use of OPTCD=FULL requires that you use OPTCD=TIMEOUT.

Default: NOFULL

OPTCD = NOTIMEOUT | TIMEOUT

Indicates completion processing for this request based on time.

If the option is OPTCD=NOTIMEOUT, this request will not be timed.

If the option is OPTCD=TIMEOUT, this request will be completed at the end of a specified time, regardless of the amount of data available.

Note: Use of OPTCD=TIMEOUT requires that you use OPBUF and OPLEN to specify the timeout option. However, use of OPTCD=TIMEOUT does not require that you use OPTCD=FULL

Default: NOTIMEOUT

OPTCD = MBUF | NOMBUF

Indicates that the DABUF parameter is the address of a Cisco IOS for S/390 MBUF, rather than a data buffer or indirect buffer list.

If the OPTCD=MBUF, review the macro using the parameter to determine how the option code will be processed.

If the option is OPTCD=NOMBUF, the DABUF parameter is processed normally.

Note: This OPTCD is for Interlink internal use only.

Default: NOMBUF

FNCCD = *function_code*

Indicates which API function to execute. The value specified must be selected from this list:

TACCEPT - Accept connection request
TADDR - Get protocol address
TBIND - Bind local protocol address
TCLEAR - Clear disconnect indication
TCLOSE - Close endpoint
TCONFIRM - Receive connect confirmation
TCONNECT - Initiate connection request
TDISCONN - Initiate abortive disconnect
TINFO - Get transport protocol information
TLISTEN - Listen for connect indications
TOPTION - Endpoint option management
TRECV - Receive from connected transport user
TRECVERR - Receive datagram error indication
TRECVFR - Receive a datagram
TREJECT - Reject connection request
TRELACK - Acknowledge orderly release indication
TRELEASE - Initiate or complete orderly release
TRETRACT - Retract a pending TLISTEN request
TSEND - Send to connected transport user
TSENDTO - Send a datagram
TUNBIND - Unbind local protocol address
TUSER - Associate user with endpoint

If a function code is specified, the definition and use of other operands is determined by the designated function. The operands that may be coded, and the rules that apply, are the same as those defined for the API macro instruction that corresponds to the function code. If a function code is not specified, the value already stored in the TPL designates the function to be executed.

Default: not indicated (to be provided later)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the transport service request associated with this TPL has been completed. The ECB can be any fullword of storage aligned on a fullword boundary.

The ECB and EXIT operands share the same storage location in the TPL, and are therefore mutually exclusive. If asynchronous mode has been specified (OPTCD=ASYNC), the ECB/EXIT field of the TPL (TPLECBXR) is used in this manner:

If ECB=address is specified, then the API uses the field as the address of an external ECB. The application program is responsible for issuing a TCHECK macro instruction to check and clear this ECB.

If EXIT=address is specified, then the API uses the field as the address of the TPL exit routine, and schedules the routine as indicated in the following EXIT operand description.

If ECB=INTERNAL is specified, then the API uses the field as an internal ECB. The application program must issue a TCHECK macro instruction to check and clear this ECB.

If synchronous mode has been specified (OPTCD=SYNC), the TPL is flagged to be processed as if ECB=INTERNAL had been specified, and the ECB/EXIT field is used as an internal ECB, which is checked and cleared automatically.

The API clears an internal ECB when it begins processing any TPL-based macro instruction, and also when the TPL is checked. However, an external ECB is only cleared when the TPL is checked. An application program using external ECBs must be sure that the external ECB is cleared before the next TPL-based macro instruction is issued.

For more information about asynchronous processing, read Cisco IOS for S/390 Assembler API Concepts, which discusses synchronization and exceptional event handling.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of a routine to be scheduled when the request represented by this TPL is completed. The EXIT and ECB operands share the same storage location in the TPL, and are therefore mutually exclusive.

The TPL exit routine is scheduled only if asynchronous mode has been indicated by OPTCD=ASYNC. If synchronous mode has been indicated, the exit routine is not used. If one is specified with this operand, the address is overwritten with an internal ECB before the request completes. For further information on asynchronous processing, read *Cisco IOS for S/390 Assembler API Concepts*.

Default: not indicated (no TPL exit routine)

MF = (L | M, [*tpl_address*]) Indicates the list or modify form of the TPL macro instruction. The second sublist operand, tpl_address, is the address of a storage area that contains the Transport Service Parameter List (TPL). If the TPL address is not provided, or the MF operand is not coded, the TPL is generated in line with the macro instruction. If the generate or execute form the TPL macro instruction is desired, the TEXEC macro instruction should be used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: not indicated (nonreentrant, in-line list)

Completion Information

If the MF operand is not coded, or MF=L is indicated, the TPL is generated at assembly time, and no executable code is expanded.

Otherwise, the macro instruction expansion contains executable code to generate or modify the TPL. If the macro instruction completes successfully, the general return code in register 15 is set to 0 (TROKAY); otherwise, the general return code is set to 12 (TRFATLPL). The function code is returned in register 0, and the TPL address is returned in register 1.

Return Codes

This table lists the symbolic names for the TPL return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-37 TPL Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code
TROKAY	func. code	n/a
TRFATLPL	func. code	n/a

Usage Information

The TPL macro instruction is used to generate or modify a Transport Service Parameter List (TPL). The TPL may be generated in line with the macro instruction, or remotely in a storage area indicated by the MF operand. The TPL macro instruction is complementary with the TEXEC macro instruction, which is used to execute a TPL.

Any operand that can be specified on other TPL-based macro instructions (except TOPEN) can be specified on the TPL macro instruction. However, this macro instruction does not check for consistency between operands used by different functions.

Example

QLSTN and NEWEP can be specified on the same TPL macro instruction, although QLSTN is used by TBIND, and NEWEP is used by TACCEPT. However, QLSTN and SEQNO can not be specified together since they both occupy the same location in the TPL. If the latter were attempted, the TPL macro instruction would generate an error at assembly time.

The TPL macro instruction is typically used to generate a TPL that is not specific to a particular function. No function-dependent processing is performed by the TPL macro instruction, even if the FNCCD operand is coded. It is generally advisable to use the list and modify forms of the appropriate TPL-based macro instructions when function-specific parameter lists are required.

The TPL macro instruction can generate a parameter list for any API TPL-based service request, except TOPEN.

Example

The macro instruction TPL QLSTN=5,FNCCD=TBIND,MF=(L,BINDTPL) generates the same parameter list as QLSTN=5,MF=(L,BINDTPL) for the TBIND macro.

The TPL macro instruction is run-to-completion, and a TCHECK macro instruction should never be issued.

TRECV

Receive Normal or Expedited Data on a Connection

The TRECV macro instruction is used to receive normal or expedited data arriving at an endpoint from the connected (or associated) transport user. TRECV is normally used to receive data on an endpoint operating in connection mode, but when supported by the transport provider, datagrams may be received at an endpoint operating in connectionless-mode if the application program has created an association with the transport user.

[*symbol*] TRECV [EP = *endpoint_id*]
[,DALEN = *user_data_length*]
[,DABUF = *user_data_address*]
[,DAALET = *user_data_alet*]
[,OPBUF = *protocol_options_address*]
[,OPLEN = *protocol_options_length*]
[,OPALET = *protocol_options_alet*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,BLOCK | NOBLOCK]
[,DIRECT | INDIR]
[,NOFULL | FULL]
[,NOTIMEOUT | TIME OUT]
[,MBUF | NOMBUF])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TRECV macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

DALEN = *user_data_length*

Indicates the length (in bytes) of a data storage area or an indirect data vector identified by the DABUF operand. The length is updated when the request is completed to reflect the actual amount of data received. If the value indicated for DALEN is zero, no data is returned to the application program.

Default: 0 (return no user data)

DABUF = *user_data_address* Indicates the address of a storage area for receiving data that has arrived at the endpoint.

If the data mode is direct, then the value specified is the address of a contiguous storage area for receiving the data.

If the data mode is not direct, then the value specified must be the address of an indirect data vector, and each element of the vector must have been initialized to point to an individual segment of non-contiguous storage.

All available data is moved into the storage area, and the length of the storage area is updated to indicate the amount of data received. If more data is available than fits in the storage area provided, the storage area is filled, and the remaining data is held by the transport provider until another TRECV macro instruction is issued.

All user data received is application-dependent, and is not interpreted by the API or the transport provider. The maximum amount of data that can be received with a single TRECV macro instruction is provider-dependent, and can be determined with the TINFO macro instruction. The storage area can be aligned on any boundary convenient for the application program.

Default: 0 (no user data storage area)

DAALET = *user_data_alet* Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the DABUF parameter. The DAALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The DAALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

OPLEN = *protocol_options_length* Indicates the length (in bytes) of the protocol option list identified by the OPBUF operand. A value of zero indicates there is no protocol option list.

Default: 0 (no protocol option list)

OPBUF = *protocol_options_address* Indicates the address of a storage area containing a protocol option list. The area must contain a list of variable-length protocol options, with each option identified by its length and name. Each entry in the list must also contain room for an options value.

The type, number, and format of protocol options is provider-dependent and the maximum size of the option list can be determined by issuing a TINFO macro instruction. The storage area can be aligned on any boundary convenient to the application program.

Default: 0 (no protocol option list)

Note The only option currently supported on the TRECV statement is TPOPRTIM.

OPALET = *protocol_options_alet* Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the OPBUF parameter. The OPALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The OPALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TRECV macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode and control is returned immediately after scheduling the TRECV request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = BLOCK | NOBLOCK

Indicates whether or not the issuing task can be suspended if the TRECV macro instruction cannot be completed immediately.

If the option is OPTCD=BLOCK(SYNC is also set, and no data is available to be received), the issuing task is suspended until more data is received at the endpoint.

If the option is OPTCD=NOBLOCK, the TRECV macro instruction is completed immediately, and an abnormal return code indicates that the task would have been suspended for an indefinite period of time.

In either case, if data is available to be received, the TRECV macro instruction completes normally without suspending the issuing task. When OPTCD=NOBLOCK is indicated, the TRECV macro instruction can be used to poll for available data. If user data has already been received, and is available, the request is complete as usual. Otherwise, the request is completed abnormally, and the transport user can try again after delaying an appropriate period of time.

Default: BLOCK (suspend issuing task if necessary)

OPTCD = DIRECT | INDIR

Indicates the format of the user data parameter.

If the option is OPTCD=DIRECT, the DABUF and DALEN operands identify a storage area into which data should be received directly.

If the option is OPTCD=INDIR, the storage area identified by these operands contains an indirect data vector. An indirect data vector consists of a list of address-length pairs, with each element identifying a separate segment of non-contiguous storage. In this case, DABUF is the address of the first element in the list, and DALEN is the total length of the list. The length of the vector must be a multiple of eight, and the total amount of data that can be received is the sum of the lengths of each data segment.

Default: DIRECT (receive directly into data area)

OPTCD = NOFULL | FULL

Indicates completion processing for this request based on the amount of data.

If the option is OPTCD=NOFULL, this request is completed as soon as any data arrives.

An option of OPTCD=FULL indicates that this request is not to be completed until either a specified timeout occurs, or the requested amount of data arrives to fill up this request.

Note: Use of OPTCD=FULL requires that you use OPTCD=TIMEOUT.

Default: NOFULL

OPTCD = NOTIMEOUT | TIMEOUT

Indicates completion processing for this request based on time.

If the option is OPTCD=NOTIMEOUT, this request will not be timed.

If the option is OPTCD=TIMEOUT, this request will be completed at the end of a specified time, regardless of the amount of data available.

Note: Use of OPTCD=TIMEOUT requires that you use OPBUF and OPLEN to specify the timeout option. However, use of OPTCD=TIMEOUT does not require that you use OPTCD=FULL.

Default: NOTIMEOUT

OPTCD = MBUF | NOMBUF

Indicates how the DABUF parameter is handled.

If the option is OPTCD = MBUF, the DABUF parameter is ignored and the MBUF address is placed in TPLDABUF upon the completion of the request.

If the option is OPTCD = NOMBUF, the DABUF parameter is processed normally.

Note: This OPTCD is for Interlink internal use only.

Default: NOMBUF

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TRECV macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TRECV macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TRECV macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TRECV macro instruction completes normally when data is available at the endpoint, and has been moved into the storage area provided by the application program. The length of the storage area is updated to indicate the amount of data received.

This table lists the options set in the TPL OPTCD field on return to the application program:

Table 1-38 describes the TPL OPTCD Field Options on TRECV Return.

Table 1-38 TPL OPTCD Field Options On TRECV Return

NOTEOM	Indicates there is more data, and the current transport service data unit must be received with multiple TRECV macro instructions. If the transport provider does not support this concept of a TSDU, NOTEOM is not indicated.
EOM (NOTEOM not set)	The last byte of data received corresponds to the end of the TSDU.

Table 1-38 TPL OPTCD Field Options On TRECV Return (Continued)

MORE	More data is buffered for the endpoint, regardless of whether the transport provider supports the concept of a TSDU. The data may be a continuation of the current TSDU, or the beginning of the next TSDU.
NOMORE (MORE not set)	No data is buffered.
EXPEDITE	The data is expedited data.
NORMAL (EXPEDITE not set)	The data is normal data.

NOTEOM and MORE apply to expedited data in the same way they apply to normal data. If NOTEOM and EXPEDITE are indicated together, the data is the beginning or continuation of an expedited transport service data unit that must be received with multiple TRECV macro instructions. If EOM and EXPEDITE are indicated, the data comprises the end of an ETSDU.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY). The conditional completion code in register 0 is always 0 (TCOKAY), and the TPL return code field is set accordingly. No other information is returned.

If the TRECV macro instruction completes abnormally, no data is moved into the storage area, and the OPTCD indicators are not set. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TRECV return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-39 TRECV Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY	TCTIME	
TRFAILED	TAINTEG	TEPROTO	TEDISCON	TERELESE
	TAENVIRO	TESYSERR TEUNSUPF	TESUBSYS TESTOP	TEDRAIN TETERM
	TAFORMAT	TEBDFNCD TEBDDATA	TEBDOPCD TEBDEXIT	TEBDECB TEBDOPTN
	TAPROCED	TEAMODE TEBUFOVR	TESTATE	TEREQOVR
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		

Table 1-39 TRECV Return Codes (Continued)

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.

Usage Information

The TRECV macro instruction is used to receive normal or expedited data that has arrived at an endpoint. The data may be part of a byte stream or message stream sent over a connection, or part (or all) of a datagram received via an association with the remote transport user.

If the transport service type is a connection-mode byte stream, data buffered at the endpoint is moved into the storage area provided by the application program. If the total amount of buffered data is less than the size of the storage area, all data is moved and the length of the storage area is updated to reflect the actual amount of data moved. NOMORE is also set in the TPL to indicate no more data is buffered at the endpoint. Otherwise, the storage area is filled to capacity, and MORE is set to indicate that more data is ready to be received.

If the transport service type is a connection-mode message stream, processing is similar. However, the transport provider must preserve the logical boundary of TSDUs, and indicate where the boundaries occur. If the data moved into the storage area comprises the end of the TSDU, EOM is set in the TPL to indicate that the TSDU has been completely received. Otherwise, NOTEOM is set to indicate another TRECV macro instruction should be issued to receive the end of the TSDU.

If the buffered data contains the end of one TSDU and the beginning of the next TSDU, only data up to the end of the current TSDU is moved into the storage area. In other words, data from two TSDUs is never moved into the storage area at one time. Another TRECV macro instruction must be executed to receive the next TSDU. MORE is set to indicate another TSDU is available to be received.

If the transport service type is connectionless-mode using associations, the datagram is handled like a TSDU. If the entire datagram does not fit within the storage area, NOTEOM is set in the TPL indicating that another TRECV macro instruction must be issued to receive the continuation of the datagram. If the data moved contains the end of the datagram, and another datagram is available at the endpoint, MORE is set to indicate a new datagram exists.

If the transport provider supports expedited data, the same macro instruction is used to receive it. TOEXPDTE is set in the TPL to distinguish normal data from expedited data. In a manner similar to the handling of TSDUs, normal and expedited data is never mixed. Datagrams are always classified as normal data.

The application program does not ask to receive expedited data; rather, it requests to receive data, and the API indicates with the TOEXPDTE flag whether the data received was expedited or normal. Also, the distinction between normal and expedited data is provider-dependent. Some transport providers can process expedited data out-of-band, and if normal and expedited data are available at the same time, expedited data is delivered ahead of normal data. However, an expedited TSDU can never interrupt a normal TSDU; that is, a normal TSDU must be received in its entirety before the expedited TSDU can be received. Transport providers that do not support out-of-band expedited data must deliver the data in sequence (for example, TCP urgent data). In this case, the application program should use the indication to expedite processing of the data stream.

TCP Provider Session Options

These options are valid only for TCP provider sessions. Each of these values is four bytes long, so the option length must be eight. Refer to TOPTION for format information.

- If the option is TPOPRTIM, the time, in seconds, to wait for data to arrive to satisfy this request. This option is valid only when specified on a TRECV TPL.

Note TPOPRTIM *must* be specified with OPTCD=TIMEOUT. It is possible for a TRECV to complete with zero bytes of data, or with less than a full request if used with OPTCD=FULL. The return values of TAOKAY and TCTIME indicate this situation.

Return Indicators

The OPTCD field of the TPL is used by the API data transfer routines to return indicators that can be tested by the application program. The indicators returned by the TRECV macro instruction correspond to the indicators set by the TSEND macro instruction. These indicators are located in the function-specific option code field mapped by the TPL dsect as TPLOPCD2.

These bits are used by the TRECV macro instruction:

- If set, the TONOTEOM bit corresponds to OPTCD=NOTEOM, and indicates that another TRECV macro instruction must be issued to receive the end of the TSDU.

  If not set, the TONOTEOM bit corresponds to OPTCD=EOM, and indicates the last byte of data moved into the storage area corresponds to the end of the TSDU.

- If set, the TOMORE bit corresponds to OPTCD=MORE and indicates that more data is buffered at the endpoint. The data is not necessarily part of the current TSDU.

  If not set, the TOMORE bit corresponds to OPTCD=NOMORE, and indicates all of the data buffered at the endpoint has been moved into the application program's storage area.

- If set, the TOEXPDTE bit corresponds to OPTCD=EXPEDITE, and indicates that the data moved into the storage area is expedited data.

  If not set, the TOEXPDTE bit corresponds to OPTCD=NORMAL, and indicates that the data moved into the storage area is normal data. If the transport provider supports the concept of an expedited transport service data unit, TONOTEOM is used to mark the continuation of the ETSDU.

The storage area provided by the application program can be a simple, contiguous segment of storage, or a set of non-contiguous segments indirectly addressed via a data vector.

- If the option is OPTCD=DIRECT, user data is transferred into the storage area identified by the DABUF and DALEN operands. The length of the storage area is updated to reflect the actual amount of data transferred.
- If the option is OPTCD=INDIR, DABUF and DALEN identify a storage area initialized with the addresses and lengths of non-contiguous storage segments into which the user data is to be transferred. The length of the indirect data vector is updated to reflect the actual amount of data transferred.

Each entry in an indirect data vector consists of a fullword address followed by a fullword length. If the length is zero, the entry is ignored. If the length is non-zero, the address must reference a valid storage area, and may be aligned on any boundary convenient for the application program. The length of the vector is used to determine the number of entries in the list. The sum of the lengths must not exceed the maximum interface data unit defined for the endpoint.

Unlike most other macro instructions, multiple TRECV macro instructions can be issued without waiting for the first to complete. However, each macro instruction requires its own TPL. The maximum number that can be issued before one must complete is an API variable that can be modified by the TOPTION macro instruction. The default value is set when the API is installed. TRECV macro instructions are completed in the order in which they are issued.

Data received with the TRECV macro instruction is buffered in the API address space before it is moved into the storage area provided by the application program. The total amount of receive buffering allocated for an endpoint is also an API option. For TLI-mode sockets, zero (0) is returned in the residual count field. For socket mode, the residual count returned when the TRECV macro instruction completes is set to the total number of bytes placed into the data buffer(s).

The data received is application-dependent and is not interpreted by the API or the transport provider. The maximum amount that can be received with a single TRECV request can be determined by issuing a TINFO macro instruction.

TRECVERR

Receive Datagram Error Indication

The TRECVERR macro instruction is used to receive an error indication associated with a datagram previously sent on an endpoint operating in connectionless-mode. A protocol-specific datagram error code is returned to the application program, as well as the remote protocol address.

[*symbol*] TRECVERR [EP = *endpoint_id*]
[,ADLEN = *protocol_address_length*]
[,ADBUF = *protocol_address_address*]
[,ADALET = *protocol_address_alet*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,TRUNC | NOTRUNC])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TRECVERR macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

ADLEN = *protocol_address_length*

Indicates the length (in bytes) of the protocol address storage area identified by the ADBUF operand. The length is updated when the request is completed to reflect the actual length of the protocol address returned. If the length is zero, the protocol address of the remote transport user is not returned to the application program.

Default: 0 (return no protocol address)

ADBUF = *protocol_address_address*

Indicates the address of a storage area for returning the protocol address of the remote transport user. The storage area should be large enough to contain the entire address. The format of the protocol address is provider-dependent, and its maximum size can be determined by issuing a TINFO macro instruction. The storage area can be aligned on any boundary.

Default: 0 (no protocol address storage area)

ADALET = *protocol_address_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the ADBUF parameter. The ADALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The ADALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=Ior MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TRECVERR macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TRECVERR request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = TRUNC | NOTRUNC

Indicates whether the protocol address or options returned to the application program by the transport provider should be truncated if they do not fit within the storage area provided.

If the option is OPTCD=TRUNC, the excess is truncated, and the TRECVERR macro instruction is completed conditionally as long as no other errors occur.

If the option is OPTCD=NOTRUNC, nothing is placed in the storage area, and the TRECVERR macro instruction is completed abnormally.

Default: NOTRUNC (no truncation)

ECB = INTERNAL | *event_control_block_address*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TRECVERR macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TRECVERR macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TRECVERR macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TRECVERR macro instruction completes normally (or conditionally) when the information associated with a datagram error indication has been moved into the storage areas provided by the application program. The protocol address of the remote transport user is returned, and the storage length is updated to reflect the amount of information returned. A protocol-dependent error code is also returned in the DGERR field of the TPL associated with this request.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY), and a conditional completion code is returned in register 0. TCTRUNC is set if the protocol address returned to the application program was truncated to fit in the storage area provided. The TPL return code field is set accordingly. No other information is returned.

If the TRECVERR macro instruction completes abnormally, no information is returned to the application program, and the datagram error indication (if any) remains pending. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TRECVERR return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-40 TRECVERR Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY	TCTRUNC	
TRFAILED	TAINTEG	TEOVRFLO		
	TAENVIRO	TESYSERR TESTOP TEUNSUPF	TESUBSYS TETERM	TEDRAIN TEUNSUPO
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD TEBADDR	TEBDECB
	TAPROCED	TEAMODE	TESTATE	TENOERR
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TRECVERR macro instruction is used to receive information associated with an error that occurred with a previously sent datagram. The transport provider returns the remote protocol address and options associated with the datagram, and a protocol-dependent error code that identifies the specific error.

The error occurred after being sent with a TSENDTO macro instruction that completed normally. When the error is detected, the transport provider generates a datagram error indication. The TRECVERR macro instruction clears the pending indication and receives the information associated with the error.

TRECVFR

Receive a Datagram

The TRECVFR macro instruction is used to receive datagrams arriving at an endpoint operating in connectionless-mode. The user data and the remote protocol address of the sender are returned to the application program.

[*symbol*] TRECVFR [EP = *endpoint_id*]
[,ADLEN = *protocol_address_length*]
[,ADBUF = *protocol_address_address*]
[,ADALET = *protocol_address_alet*]
[,DALEN = *user_data_length*]
[,DABUF = *user_data_address*]
[,DAALET = *user_data_alet*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,TRUNC | NOTRUNC]
[,BLOCK | NOBLOCK]
[,DIRECT | INDIR]
[,MBUF | NOMBUF])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TRECVFR macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

ADLEN = *protocol_address_length*

Indicates the length (in bytes) of the protocol address storage area identified by the ADBUF operand. The length is updated when the request is completed to reflect the actual length of the protocol address returned. If the length is zero, the protocol address of the sending transport user is not returned to the application program.

Default: 0 (return no protocol address)

ADBUF = *protocol_address_address*

Indicates the address of a storage area for returning the protocol address of the sending transport user. The storage area should be large enough to contain the entire address. The format of the protocol address is provider-dependent, and its maximum size can be determined by issuing a TINFO macro instruction. The storage area can be aligned on any boundary.

Default: 0 (no protocol address storage area)

ADALET = *protocol_address_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the ADBUF parameter. The ADALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The ADALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

DALEN = *user_data_length*

Indicates the length (in bytes) of a data storage area or an indirect data vector identified by the DABUF operand. The length is updated when the request is completed to reflect the actual amount of data received. If the value indicated for DALEN is zero, no data is returned to the application program.

Default: 0 (return no user data)

DABUF = *user_data_address*

Indicates the address of a storage area for receiving data that has arrived at the endpoint.

If the data mode is direct, the value specified is the address of a contiguous storage area for receiving the data.

If the data mode is not direct, the value specified must be the address of an indirect data vector, and each element of the vector must have been initialized to point to an individual segment of non-contiguous storage.

If more data is available than fits in the storage area provided, the storage area is filled, and the remaining data is held by the transport provider until another TRECVFR macro instruction is issued. Otherwise, all available data is moved into the storage area, and the length of the storage area is updated to indicate the amount of data received.

All user data received is application-dependent, and is not interpreted by the API or the transport provider. The maximum amount of data that can be received with a single TRECVFR macro instruction is provider-dependent, and can be determined with the TINFO macro instruction. The storage area can be aligned on any boundary convenient for the application program.

Default: 0 (no user data storage area)

DAALET = *user_data_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the DABUF parameter. The DAALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The DAALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TRECVFR macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, The request is executed in asynchronous mode, and control is returned immediately after scheduling the TRECVFR request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = TRUNC | NOTRUNC

Indicates whether the protocol address returned to the application program by the transport provider should be truncated if it does not fit within the storage area provided.

If the option is OPTCD=TRUNC, the excess is truncated, and the TRECVFR macro instruction is completed conditionally as long as no other errors occur.

If the option is OPTCD=NOTRUNC, nothing is placed in the storage area, and the TRECVFR macro instruction is completed abnormally.

Default: NOTRUNC (no truncation)

OPTCD = BLOCK | NOBLOCK

Indicates whether or not the issuing task can be suspended if the TRECVFR macro instruction cannot be completed immediately.

If the option is OPTCD=BLOCK (and no data is available to be received), the issuing task is suspended until more data is received at the endpoint.

If the option is OPTCD=NOBLOCK, the TRECVFR macro instruction is completed immediately, and an abnormal return code indicates that the task would have been suspended for an indefinite period of time.

In either case, if a datagram is available to be received, the TRECVFR macro instruction completes normally without suspending the issuing task.

When OPTCD=NOBLOCK is indicated, the TRECVFR macro instruction can be used to poll for available datagrams. If a datagram has already been received and is available, the request is complete as usual. Otherwise, the request is completed abnormally, and the transport user can try again after delaying an appropriate period of time.

Default: BLOCK (suspend issuing task if necessary)

OPTCD = DIRECT | INDIR

Indicates the format of the user data parameter.

If the option is OPTCD=DIRECT, the DABUF and DALEN operands identify a storage area into which data should be received directly.

If the option is OPTCD=INDIR, the storage area identified by these operands contains an indirect data vector. An indirect data vector consists of a list of address-length pairs, with each element identifying a separate segment of non-contiguous storage. In this case, DABUF is the address of the first element in the list, and DALEN is the total length of the list. The length of the vector must be a multiple of eight, and the total amount of data that can be received is the sum of the lengths of each data segment.

Default: DIRECT (receive directly into data area)

OPTCD = MBUF | NOMBUF

Indicates how the DABUF parameter is handled.

If the option is OPTCD=MBUF, the DABUF parameter is ignored and the MBUF address is placed in TPLDABUF upon the completion of the request.

If the option is OPTCD=NOMBUF, the DABUF parameter is processed normally.

Note: This OPTCD is for Interlink internal use only.

Default: NOMBUF

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TRECVFR macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TRECVFR macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TRECVFR macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TRECVFR macro instruction completes normally (or conditionally) when data is available at the endpoint, and has been moved into the storage area provided by the application program. The length of the storage area is updated to reflect the amount of data received with this request. A residual count is also returned which is set to the amount of internal buffer space for which no receive is pending.

If the data returned to the application program is the beginning of a new datagram, the protocol address of the sender is returned if a storage area has been provided. The corresponding lengths are updated to reflect the actual amount of data returned. If the data is the continuation of an old datagram, the lengths of these storage areas are set to zero.

On return to the application program, TONOTEOM is indicated in the field if there is more data, and the current datagram must be received with multiple TRECVFR macro instructions. If end of message is indicated in the OPTCD field (in other words, TONOTEOM not set), the last byte of the data received corresponds to the end of the datagram.

TOMORE is indicated in the OPTCD field if more data is buffered for the endpoint. The data may be a continuation of the current datagram, or the beginning of the next datagram. If no data is buffered, TOMORE is not set in the TPLOPTCD field.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY), and a conditional completion code is returned in register 0. TCTRUNC is set if the protocol address returned to the application program was truncated to fit in the storage area provided. The TPL return code field is set accordingly. No other information is returned.

If the TRECVFR macro instruction completes abnormally, no data is moved into the storage area, and the OPTCD indicators described above are not set. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TRECVFR return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-41 TRECVFR Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY	TCTRUNC	
TRFAILED	TAINTEG	TEPROTO	TEOVRFLO	
	TAENVIRO	TESYSERR TESTOP TEUNSUPF	TESUBSYS TETERM	TEDRAIN TEUNSUPO
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD TEBDADDR	TEBDECB TEBDDATA
	TAPROCED	TEAMODE TEBUFOVR	TESTATE	TEREQOVR
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TRECVFR macro instruction is used to receive a datagram that has arrived at an endpoint operating in connectionless mode. The protocol address of the sending transport user, protocol options associated with the datagram, and user data contained within the datagram itself are returned to the application program. The endpoint must be in the disabled (TSDSABLD) state when the TRECVFR macro instruction is issued.

A datagram may span several TRECVFR macro instructions if the receiving storage area is not large enough to hold the entire amount of data. If the storage area is filled to capacity, but more data remains for the current datagram, NOTEOM is set in the TPL to indicate another TRECVFR macro instruction is required to receive the continuation of the datagram. This indication continues to be

set until the end of the datagram has been received, at which time EOM is indicated. The protocol address and any protocol options associated with the datagram are returned with the first segment of the datagram.

If another datagram is available to be received, MORE is indicated at the completion of the TRECVFR macro instruction. Otherwise, NOMORE is indicated. Data from two different datagrams is never combined, even if both could fit within the storage area provided.

The TPLOPTCD field of the TPL is used by the API data transfer routines to return indicators that can be tested by the application program. The indicators returned by the TRECVFR macro instruction correspond to the indicators set by the TSENDTO macro instruction. These indicators are located in the function-specific option code field mapped by the TPL dsect as TPLOPCD2.

Table 1-42 describes Bits Used by the TRECVFR Macro Instructions

Table 1-42 Bits Used by the TRECVFR Macro Instruction

TONOTEOM	If set, corresponds to OPTCD=NOTEOM, and indicates that another TRECVFR macro instruction must be issued to receive the end of the datagram. If not set, corresponds to OPTCD=EOM, and indicates the last byte of data moved into the storage area corresponds to the end of the datagram.
TOMORE	If set, corresponds to OPTCD=MORE, and indicates that more data is buffered at the endpoint. The data is not necessarily part of the current datagram. If not set, corresponds to OPTCD=NOMORE, and indicates all of the data buffered at the endpoint has been moved into the application program's storage area.

The storage area provided by the application program can be a simple, contiguous segment of storage, or a set of non-contiguous segments indirectly addressed via a data vector.

- If the option is OPTCD=DIRECT. The datagram is transferred into the storage area identified by the DABUF and DALEN operands. The length of the storage area is updated to reflect the actual amount of data transferred.
- If the option is OPTCD=INDIR, the DABUF and DALEN operands identify a storage area initialized with the addresses and lengths of non-contiguous storage segments into which the datagram is to be transferred. The length of the indirect data vector is updated to reflect the actual amount of data transferred.

Each entry in an indirect data vector consists of a fullword address followed by a fullword length. If the length is zero, the entry is ignored. If the length is non-zero, the address must reference a valid storage area, and may be aligned on any boundary convenient for the application program. The length of the vector is used to determine the number of entries in the list. The sum of the lengths must not exceed the maximum interface data unit defined for the endpoint.

Unlike most other macro instructions, multiple TRECVFR macro instructions can be issued without waiting for the first to complete. However, each macro instruction requires its own TPL. The maximum number that can be issued before one must complete is an API variable that can be modified by the TOPTION macro instruction. The default value is set when the API is configured. TRECVFR macro instructions are completed in the order in which they are issued.

Data received with the TRECVFR macro instruction is buffered in the API address space before it is moved into the storage area provided by the application program. The total amount of receive buffering allocated for an endpoint is also an API option.

The data received is application-dependent, and is not interpreted by the API or the transport provider. The maximum amount that can be received with a single TRECVFR request can be determined by issuing a TINFO macro instruction.

TREJECT

Reject a Connection Request

When a connect indication has been received at an endpoint with a TLISTEN macro instruction, the TREJECT macro instruction is used to reject the connection request.

[*symbol*] TREJECT [EP = *endpoint_id*]
[,SEQNO = *sequence_number*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,ABORT | CLEAR])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E,[*tpl_address*])]

Syntax Description

EP = *endpoint_id*
Specifies the endpoint at which the TREJECT macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

SEQNO = *sequence_number*
Specifies which connect indication is being rejected. The value specified must have been returned by a TLISTEN macro instruction. The transport provider uses this value to identify a connect indication pending for this endpoint, which has not yet been accepted or rejected.

Default: 0 (most likely an invalid sequence number)

OPTCD = SHORT | LONG | **EXTEND**
Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TREJECT macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TREJECT request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = ABORT | CLEAR

Indicates the action to be taken by the transport provider when the application program issues a TREJECT macro instruction at an endpoint for which a disconnect indication is already pending.

If the option is OPTCD=ABORT, the request is completed abnormally, and the application program must clear the pending disconnect indication with a TCLEAR macro instruction.

If the option is OPTCD=CLEAR, the TREJECT request clears the disconnect indication, and the macro instruction is completed normally (or conditionally) if no other errors occur.

Default: CLEAR (clear pending disconnect indication)

ECB = INTERNAL | event_control_block_addr

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TREJECT macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TREJECT macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TREJECT macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TREJECT macro instruction completes normally when the disconnect protocol data unit has been scheduled by the transport provider for transmission to the calling transport user, and the pending connect indication has been removed from the queue. The state of the endpoint is changed from connect-indication-pending (TSINCONN) to enabled (TSENABLD) if no other indications are pending. Otherwise, the state of the endpoint is unchanged.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY). The conditional completion code in register 0 is always 0 (TCOKAY), and the TPL return code field is set accordingly. No other information is returned.

If the TREJECT macro instruction completes abnormally, no disconnect protocol data unit is scheduled for transmission to the calling transport user, and the connect indication remains pending. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TREJECT return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-43 TREJECT Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY		
		TAINTEG	TEPROTO	TEDISCON
	TAENVIRO	TESYSERR TESTOP TEUNSUPF	TESUBSYS TETERM	TEDRAIN TEUNSUPO
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD TEBDDATA	TEBDECB TEBDSQNO
	TAPROCED	TEAMODE	TESTATE	TEINCMPL
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TREJECT macro instruction is used to refuse a connection request received from a remote transport user.

The TREJECT macro instruction is issued in place of a TACCEPT macro instruction to reject a pending connect indication, and to remove it from the queue. The sequence number returned with a TLISTEN macro instruction must be provided to designate which connect indication is being rejected. The endpoint must be in the connect-indication-pending (TSINCONN) state, and remains in this state if more connect indications are pending. Otherwise, the endpoint is returned to the enabled (TSENABLD) state, and remains enabled to receive more connect indications.

The TREJECT macro instruction is usually executed at endpoints operating in connection mode. However, if a TLISTEN macro instruction has been executed at an endpoint operating in connectionless mode, the TREJECT macro instruction may be issued to reject a simulated connect indication. The connect indication was generated as the result of receiving a datagram at the endpoint, and the TREJECT macro instruction causes the indication to be removed, and the associated datagram to be discarded. No association is established, and another TLISTEN macro instruction should be issued to receive the next indication when a datagram arrives.

TRELACK

Acknowledge Orderly Release Indication

The TRELACK macro instruction is used to acknowledge an indication of an orderly release request generated at an endpoint normally operating in connection-mode. The application program can no longer receive data, but is allowed to continue sending data until the connection is fully released by issuing a TRELEASE macro instruction.

[*symbol*] TRELACK [EP = *endpoint_id*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,BLOCK | NOBLOCK]
[,ABORT | CLEAR])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TRELACK macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TRELACK macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode and control is not returned to the application program until the requested macro instruction completes. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode and control is returned immediately after scheduling the TRELACK request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = ABORT | CLEAR

Indicates the action to be taken by the transport provider when the application program issues a TRELACK macro instruction at an endpoint for which a disconnect indication is already pending.

If the option is OPTCD=ABORT, the request is completed abnormally, and the application program must clear the pending disconnect indication with a TCLEAR macro instruction.

If the option is OPTCD=CLEAR, the TRELACK request clears the disconnect indication, and the macro instruction is completed normally (or conditionally) if no other errors occur.

Default: CLEAR (clear pending disconnect indication)

OPTCD = BLOCK | NOBLOCK

Indicates whether or not the issuing task can be suspended if the TRELACK macro instruction cannot be completed immediately.

If the option is OPTCD=BLOCK (and no release indication is pending), the issuing task is suspended until an orderly release is received.

If the option is OPTCD=NOBLOCK, the TRELACK macro instruction is completed immediately, and an abnormal return code indicates that the task would have been suspended for an indefinite period of time.

In either case, if an orderly release indication is already pending, the TRELACK macro instruction completes immediately without suspending the issuing task.

When OPTCD=NOBLOCK is indicated, the TRELACK macro instruction can be used to poll for a release indication. If a release indication is pending, the request is completed as usual. Otherwise, the request is completed abnormally, and the transport user can try again after delaying an appropriate period of time.

Default: BLOCK (suspend issuing task if necessary)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TRELACK macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TRELACK macro associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF= (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TRELACK macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TRELACK macro instruction completes normally when an orderly release indication as been received.

- If the state of the endpoint was connected (TSCONNCT), the state is changed to release-indication-pending (TSINRLSE).
- If the state of the endpoint was not connected, the state must have been release-in-progress (TSOURLSE), and is changed to disabled (TSDSABLD) or enabled (TSENABLD), depending on the operating mode of the transport user.

The connection is not released until the endpoint returns to the disabled or enabled state.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY). The conditional completion code in register 0 is always 0 (TCOKAY), and the TPL return code field is set accordingly. No other information is returned.

If the TRELACK macro instruction completes abnormally, the release indication (if any) remains queued. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TRELACK return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-44 TRELACK Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY		
TRFAILED	TAINTEG	TEPROTO	TEDISCON	
	TAENVRO	TESYSERR TESTOP	TESUBSYS TEUNSUPF	TEDRAIN

Table 1-44 TRELACK Return Codes (Continued)

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD	TEBDECB
	TAPROCED	TEAMODE TENORLSE	TESTATE	TEINCMPL
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TRELACK macro instruction is used to acknowledge an orderly release indication that has been generated at an endpoint. It is used in conjunction with the TRELEASE macro instruction to terminate a connection without loss of data.

An orderly release indication is generated by an orderly release protocol data unit arriving at an endpoint. When all of the data buffered at the endpoint has been received by the application program, the release indication is generated. The application must issue a TRELACK macro instruction to acknowledge the indication.

- If the release was initiated by the remote transport, the application may continue sending data until a TRELEASE macro instruction is executed.
- If the release was initiated by the application program, acknowledgment of the release indication causes the connection to be terminated.

The orderly release procedure is primarily intended for connection-mode operation. However, if supported by the transport provider, associations established in connectionless mode can be terminated using the orderly release procedure. If the application program issues a TRELEASE macro instruction, a release indication is generated as soon as all inbound data buffered at the endpoint has been received. The orderly release service is simulated by the API, and is transparent to the transport provider.

TRELEASE

Initiate Orderly Release

The TRELEASE macro instruction is used to initiate or complete the orderly release of a connection. This macro instruction provides a graceful termination of a connection and is not immediate as is the abortive disconnect initiated with the TDISCONN macro instruction. Any data previously sent with a TSEND macro instruction is delivered.

[*symbol*] TRELEASE [EP = *endpoint_id*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TRELEASE macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TRELEASE macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TRELEASE request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TRELEASE macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TRELEASE macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TRELEASE macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TRELEASE macro instruction completes normally when the orderly release protocol data unit has been scheduled for transmission to the connected transport user.

- If the state of the endpoint was connected (TSCONNCT), the state is changed to release-in-process (TSOURLSE).
- If the state of the endpoint was not connected, the state must have been release-indication-pending (TSINRLSE), and is changed to disabled (TSDSABLD) or enabled (TSENABLD), depending on the operating mode of the application program.

The connection is not released until the endpoint returns to the disabled or enabled state.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY). The conditional completion code in register 0 is always 0 (TCOKAY), and the TPL return code field is set accordingly. No other information is returned.

If the TRELEASE macro instruction completes abnormally, no orderly release protocol data unit is scheduled for transmission. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TRELEASE return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-45 TRELEASE Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY		
TRFAILED	TAINTEG	TEPROTO	TEDISCON	
	TAENVIRO	TESYSERR TESTOP	TESUBSYS TETERM	TEDRAIN TEUNSUPF
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD	TEBDECB
	TAPROCED	TEAMODE	TESTATE	TEINCMPL
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TRELEASE macro instruction is used to initiate or complete the orderly release of a connection. It differs from the TDISCONN macro instruction in that it is not immediate, and the connection is not released until both transport users agree to do so. Also, any unsent data buffered at the endpoint is forwarded to the peer transport user. The TRELEASE macro instruction does not complete until all pending TSEND macro instructions have completed.

The orderly release procedure requires both transport users to request orderly release before the connection is terminated. If the application program initiates orderly release by executing a TRELEASE macro instruction, all buffered data is forwarded to the peer transport user, followed by an orderly release request. The application program should then continue receiving data until an orderly release indication is received. Any attempt to send data through an endpoint in the release-in-progress state completes abnormally.

If the peer transport user initiates an orderly release, a release indication is generated and presented to the application program. The application program should cease receiving data, and execute a TRELACK macro instruction to acknowledge the release indication. The application program can continue sending data until transmission is complete, and then a TRELEASE macro instruction should be issued to complete termination of the connection.

Orderly release may not be supported by the transport provider, and should be requested when the endpoint is opened (see TOPEN). If not requested in advance, the TINFO macro instruction should be issued to determine if orderly release is supported. If not, the application program should use a session protocol, or some other method to terminate the session with the peer transport user, to avoid losing data that may be discarded by an abortive disconnect.

The TRELEASE macro instruction is generally executed at endpoints operating in connection mode. If an association has been established for an endpoint operating in connectionless mode, the API simulates the orderly release procedure, and generates a release indication after all data has been sent to the peer transport user. Note however, there is no guarantee that the data has been received. This procedure is transparent to the transport provider.

TRETRACT

Retract Pending Listen Request

The TRETRACT macro instruction is used to retract a pending listen initiated with the TLISTEN macro instruction. Any outstanding listen is forced to complete abnormally, and the state of the endpoint is as if the TLISTEN macro instruction had not been executed at all.

[*symbol*] TRETRACT [EP = *endpoint_id*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E, [*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TRETRACT macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TRETRACT macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TRETRACT request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TRETRACT macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TRETRACT macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TRETRACT macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TRETRACT macro instruction completes normally when a TLISTEN request, which was pending at the endpoint, is prematurely terminated. The pending TLISTEN request is completed abnormally, and the specific error code is set to TERETRCT to indicate that a TRETRACT macro instruction forced its completion. The state of the endpoint is unchanged, and is as if the TLISTEN macro instruction had not been executed. If no connect indications were pending at the time the listen was retracted, the endpoint is left in the enabled (TSENABLD) state. Otherwise, it is left in the connect-indication-pending (TSINCONN) state.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY). The conditional completion code in register 0 is always 0 (TCOKAY), and the TPL return code field is set accordingly. No other information is returned.

If no TLISTEN request was pending, perhaps because it had already completed, the TRETRACT macro instruction is completed abnormally. The general return code in register 15, and the recovery action code in register 0, indicate the nature of the failure. The TPL return code field may contain a specific error code which identifies the particular error.

Return Codes

This table lists the symbolic names for the TRETRACT return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-46 TRETRACT Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY		
TRFAILED	TAEXCPTN	TENOLSTN		
	TAENVIRO	TESYSERR TESTOP	TESUBSYS TETERM	TEDRAIN
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD	TEBDECB
	TAPROCED	TEAMODE	TESTATE	TEINCMPL
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TRETRACT macro instruction can be used to retract a pending TLISTEN macro instruction by forcing it to complete abnormally. Once the TLISTEN request has completed, the protocol address can be unbound from the endpoint. Until then, the endpoint can continue receiving connect indications. TRETRACT provides an alternative to closing the endpoint in order to prevent it from listening for connect indications. If any connect indications are pending at the endpoint, they remain pending, and must be accepted or rejected before the protocol address may be unbound.

There is no guarantee that the TLISTEN request can be retracted before it would be completed normally by an incoming connection request. If this happens, the TRETRACT function does not find any record of a pending TLISTEN request, and completes the TRETRACT macro instruction abnormally. If the TRETRACT macro instruction was issued too late to prevent TLISTEN from completing normally, the pending connect indication must be processed as usual by accepting or rejecting it with the appropriate macro instruction.

TSEND

Send Normal or Expedited Data on a Connection

The TSEND macro instruction is used to send normal or expedited data to the peer transport user connected to an endpoint. TSEND is normally used to send data on an endpoint operating in connection mode, but when supported by the transport provider, datagrams may be sent on an endpoint operating in connectionless-mode if the application program has established an association with the transport user.

[*symbol*] TSEND [EP = *endpoint_id*]
[,DALEN = *user_data_length*]
[,DABUF = *user_data_address*]
[,DAALET = *user_address_alet*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,MORE | NOMORE]
[,NORMAL | EXPEDITE]
[,EOM | NOTEOM]
[,DIRECT | INDIR]
[,BLOCK | NOBLOCK]
[,MBUF | NOMBUF])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E,[*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TSEND macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

DALEN = *user_data_length*

Indicates the length (in bytes) of a data storage area or an indirect data vector identified by the DABUF operand.

If the data mode is direct, the amount of user data to be sent is equal to the length of the storage area.

If the data mode is indirect, the total amount of user data is equal to the sum of all data segments identified by the data vector.

In either case, the total amount of user data must not exceed the limit supported by the transport provider. This limit can be obtained with the TINFO macro instruction. A length of zero indicates there is no user data to be sent.

Default: 0 (no user data)

DABUF = *user_data_address*

Indicates the address of user data to be sent to the connected (or associated) transport user.

If the data mode is direct, the value specified is the address of the storage area containing the user data.

If the data mode is indirect, the value specified must be the address of an indirect data vector, and each element of the vector must have been initialized to point to an individual segment of user data.

If no data is available, the length as indicated by the DALEN operand should be zero. The content of all user data is application-dependent, and is not interpreted by the API or the transport provider. The storage area can be aligned on any boundary convenient for the application program.

Default: 0 (no user data storage area)

DAALET = *user_address_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the DABUF parameter. The DAALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The DAALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TSEND macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TSEND request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = MORE | NOMORE

Indicates whether the application program intends to immediately send more data, or intends to pause momentarily until it has more data to send.

If the option is OPTCD=MORE, the application program expects to immediately issue another TSEND macro instruction at the same endpoint.

If the option is OPTCD=NOMORE, the application program has no more data to send, but intends to leave the connection established, and may resume sending data later.

The interpretation of this option code by the transport provider is protocol dependent. The intent is that the transport provider uses this information to augment its packet algorithm, and deduce when unsent data must be forwarded on the connection. Not all connection-mode transport providers are required to interpret this option code, but all are required to accept it. If the TSEND macro instruction is executed in synchronous mode, OPTCD=MORE is ignored.

No implication is drawn about message boundaries by the indication of OPTCD=NOMORE. If the underlying transport protocol can preserve logical boundaries within the data stream, then such boundaries should be indicated with OPTCD=EOM.

Default: NOMORE (send data immediately)

OPTCD = NORMAL | EXPEDITE

Indicates whether the user data should be sent as normal or expedited data.

If the option is OPTCD=NORMAL, the data associated with this request is to be sent as normal data.

If the option is OPTCD=EXPEDITE, the data is to be sent as expedited data.

The distinction between normal and expedited data is left to the interpretation of the transport provider.

Default: NORMAL (send data as normal data)

OPTCD = EOM | NOTEOM

Indicates whether the data associated with this request is a complete message, or is continued with one or more subsequent macro instructions.

If the option is OPTCD=EOM, the last byte of data corresponds to the end of the message or datagram.

If the option is OPTCD=NOTEOM, the end of the message or datagram does not occur with this request, and is continued with at least one more TSEND or TSENDTO macro instruction.

Although all transport providers supplying connection-mode service are required to accept this option code, only those that can preserve logical boundaries in the data stream are required to interpret it. Generally, these providers support the concept of a transport service data unit to delineate such boundaries. If the transport provider does not preserve logical boundaries, this option code is ignored. The setting of this option code implies nothing about how the user data is broken down into packets by the underlying protocol for sending to the peer transport user.

Transport providers supplying connectionless-mode service are not required to accept this option code. Those that do, interpret OPTCD=NOTEOM to mean the datagram is continued with another TSEND macro instruction. In this case, a datagram is synonymous with TSDU, except that implementation is purely a local concern.

Default: EOM (end of message or datagram)

OPTCD = DIRECT | INDIR

Indicates the format of the user data parameter.

If the option is OPTCD=DIRECT, the DABUF and DALEN operands identify a storage area into which data should be received directly.

If the option is OPTCD=INDIR, the storage area identified by these operands contains an indirect data vector. An indirect data vector consists of a list of address-length pairs, with each element identifying a separate segment of non-contiguous storage. In this case, DABUF is the address of the first element in the list, and DALEN is the total length of the list. The length of the vector must be a multiple of eight, and the total amount of data that can be received is the sum of the lengths of each data segment.

Default: DIRECT (send directly from data area)

OPTCD = MBUF | NOMBUF

Indicates that the DABUF parameter is the address of a Cisco IOS for S/390 MBUF, rather than a data buffer or indirect buffer list.

If the option is OPTCD=MBUF, the DALEN parameter is ignored and the length of the data is determined from fields within the MBUF structure.

If the option is OPTCD=NOMBUF, the DABUF parameter is processed normally.

Note: OPTCD=MBUF is intended only for internal Interlink applications and is used to improve performance.

Default: NOMBUF

OPTCD = BLOCK | NOBLOCK

OPTCD=NOBLOCK may be used with endpoints opened with MODE=SOCKETS.

This option is ignored for TLI-mode endpoints, which always block until the sent data is acknowledged. Normally, socket-mode endpoints do not block. However, if the amount of send data exceeds the amount of available buffer space, the TSEND request will block by default until buffer space becomes available. OPTCD=NOBLOCK may be used in this case to prevent the endpoint from becoming blocked. When this occurs, only the amount of data for which there is space is sent.

Buffer space is limited by configuration parameters and TOPTION negotiation.

If the option is OPTCD=NOBLOCK, then the TSEND will send either the amount of available space in the send buffer or the amount of data that was requested by the TSEND, whichever is less.

Default: BLOCK

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TSEND macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TSEND macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TSEND macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TSEND macro instruction completes normally when the data has been moved from the application program's storage area, and has been forwarded to the transport provider for sending to the connected (or associated) transport user. For MODE=SOCKETS, the count of the data bytes sent is returned in the TPLCOUNT field.

Normal completion of the TSEND macro implies nothing in regard to when the data is sent to the peer transport user, and should be interpreted to mean that the transport provider has taken custody of the user data, and the storage area provided by the application program can be reused by another TSEND macro instruction. If OPTCD=NOTEOM was indicated, no assumption should be made (unless the endpoint is operating in connectionless mode) about previous fragments of the current TSDU not being sent. If OPTCD=NOMORE was indicated, the transport provider is normally coerced into sending any buffered data, but this may not occur synchronously with the completion of the macro instruction.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY). The conditional completion code in register 0 is always 0 (TCOKAY), and the TPL return code field is set accordingly. No other information is returned.

If the TSEND macro instruction completes abnormally, no user data is sent to the peer transport user. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TSEND return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-47 TSEND Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY		
TRFAILED	TAEXCPTN	TENOBLOK		
	TAINTEG	TEPROTO	TEDISCON	
	TAENVIRO	TESYSERR TESTOP	TESUBSYS TETERM	TEDRAIN TEUNSUPF
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD TEBDDATA	TEBDECB
	TAPROCED	TEAMODE TEBUFOVR	TESTATE	TEREQOVR
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TSEND macro instruction is used to send normal or expedited data through an endpoint. The data may be part of a byte stream or message stream being sent over a connection, or part (or all) of a datagram to be sent via an association to a peer transport user.

- If the transport service type is a connection-mode byte stream, data is moved from the application program's storage areas to storage areas maintained by the transport provider, broken down into packets, and sent to the connected transport user. Logical boundaries are not preserved in the data stream. The data is delivered to the peer transport user in the precise order in which it was sent, but may be fragmented in an entirely different manner. The EOM and NOTEOM indications set in the TPL are ignored.
- If the transport service type is a connection-mode message stream, data is processed in a similar manner. However, the EOM and NOTEOM indications set in the TPL are used to delineate the boundaries of transport service data units. When the data is delivered to the peer transport user, the continuation or end of a TSDU is similarly indicated. The concept of a TSDU implies nothing about how the underlying protocol creates data packets for transmission. The maximum size of a TSDU can be determined by issuing a TINFO macro instruction.

Transport providers operating in connectionless mode are not required to accept or interpret the EOM and NOTEOM indications. If interpreted, the concept of a message (or TSDU) is synonymous with a datagram, and the significance of the EOM and NOTEOM indications is only a local phenomenon. The intent is to provide a mechanism for the application program to send a large message as multiple, noncontiguous fragments. If the transport provider supports this option code, it is required to buffer the fragments and send the message as a single datagram.

Data is not necessarily broken down into packets and sent each time a TSEND macro instruction is issued, or when a message boundary is indicated. In fact, the transport provider may intentionally delay sending data as the result of performance optimization or congestion avoidance algorithms. Generally, a continual flow of data generated by the sender causes data to be forwarded. However, in interactive applications, the transport provider may need to be coerced into forwarding any buffered data.

The MORE and NOMORE indications control this process.

- If MORE is indicated, the application intends to immediately send more data, and the transport provider is free to delay sending any data associated with the current request.
- If NOMORE is indicated, the application has no more data to send, and all buffered data should be forwarded to the peer transport user.

For example, if the underlying protocol is TCP, the NOMORE indication would cause the PUSH flag to be set.

NOMORE is ignored when the synchronization mode is ASYNC.

If the transport provider supports expedited data, the TSEND macro instruction is also used to send it. The NORMAL and EXPEDITE indicators are used to distinguish normal and expedited data. The resulting actions of the transport provider is protocol-dependent. Some transport providers support the concept of an expedited transport service data unit, and others support the concept of expedited data within a data stream without logical boundaries. The EOM and NOTEOM indications apply to an expedited TSDU in the same way they apply to a normal TSDU. The form of expedited data supported by the transport provider, and the maximum size of an ETSDU, can also be determined with the TINFO macro instruction.

The OPTCD field of the TPL is used by the application program to set indicators that can be tested by the API data transfer routines. The indicators set by the TSEND macro instruction correspond to the indicators returned by the TRECV macro instruction. These indicators are located in the function-specific option code field mapped by the TPL dsect as TPLOPCD2.

These bits are used by the TSEND macro instruction:

- If set, the TONOTEOM bit corresponds to OPTCD=NOTEOM, and indicates that another macro instruction is issued to send the continuation of the TSDU or datagram.
- If not set, the TONOTEOM corresponds to OPTCD=EOM, and indicates the last byte of data moved from the storage area corresponds to the end of the TSDU or datagram.
- If set, the TOMORE bit corresponds to OPTCD=MORE, and indicates that the application program intends to immediately issue another TSEND macro instruction to send more data. The data is not necessarily part of the current message.
- If not set, the TOMORE bit corresponds to OPTCD=NOMORE, and indicates that the transport provider should forward all data buffered at the endpoint to the peer transport user.
- If set, the TOEXPDTE bit corresponds to OPTCD=EXPEDITE, and indicates that the data contained in the storage area should be sent as expedited data.
- If not set, the TOEXPDTE bit corresponds to OPTCD=NORMAL, and indicates that the data should be sent as normal data. If the transport provider supports the concept of an expedited transport service data unit (ETSDU), TONOTEOM is used to mark the continuation of the ETSDU.

User data may be provided in a simple, contiguous segment of storage, or in a set of non-contiguous segments indirectly addressed via a data vector.

The storage area provided by the application program can be a simple, contiguous segment of storage, or a set of non-contiguous segments indirectly addressed via a data vector.

- If the option is OPTCD=DIRECT, user data must be contained in the storage area identified by the DABUF and DALEN operands.
- If the option is OPTCD=INDIR, the DABUF and DALEN operands identify a storage area initialized with the addresses and lengths of non-contiguous storage segments containing the user data. The total amount of data to be transferred is the sum of the lengths of the individual segments. The total length must not exceed the maximum size of the interface data unit supported by the transport provider, or the maximum size of a transport service data unit. Upon completion of the TSEND request, the length of the indirect data vector is updated to reflect the actual amount of data transferred.

Each entry in an indirect data vector consists of a fullword address followed by a fullword length. If the length is zero, the entry is ignored. If the length is non-zero, the address must reference a valid storage area containing user data, and may be aligned on any boundary convenient for the application program. The length of the vector is used to determine the number of entries in the list.

Unlike most other macro instructions, multiple TSEND macro instructions can be issued without waiting for the first to complete. However, each macro instruction requires its own TPL. The maximum number that can be issued before one must complete is the API variable that can be modified by the TOPTION macro instruction. The default value is set when the API is installed. TSEND macro instructions are completed in the order in which they are issued.

Data sent with the TSEND macro instruction is buffered in the API address space before it is forwarded to the transport provider. The total amount of send buffering allocated for an endpoint is also an the API option.

User data is application-dependent, and is not interpreted by the API or the transport provider. The maximum amount that can be sent with a single TSEND request can be determined by issuing a TINFO macro instruction.

Data Transfer Modes

TSEND handles data transfer according to the transfer mode specified on the TOPEN macro. TLI or SOCKET may be specified; TLI is the default.

TLI Mode

TLI is the standard data transfer mode for Cisco IOS for S/390. For TLI mode, specify MODE=TLI on the TOPEN macro, or allow it to default.

- A TSEND request in TLI mode completes when all of the data has been acknowledged by the remote TCP. The amount of data that can be sent is subject to the limits set by the LSEND and LTSND values.
- A TLI mode TSEND sends either all of the data, or none of it.
- Data contained in the application program's storage area is moved into the API address space when the TSEND macro instruction is accepted. Therefore, if the TSEND macro instruction is executed in asynchronous mode, the application program can reuse the storage area after the macro instruction completes but before the request is completed. In fact, the storage area may be used to send more data as long as an inactive TPL is available.
- TPLCOUNT is cleared to zero.

Socket Mode

Socket mode is a new data transfer mode for Version 5.2 of Cisco IOS for S/390. For socket mode, specify MODE=SOCKET on the TOPEN macro.

- A TSEND request in socket mode completes when all of the data that will be sent is passed to the local transport provider (for example, TCP or UDP).
- The amount of data that is actually sent for TSEND depends upon whether OPTCD=BLOCK or OPTCD=NOBLOCK is used. In either case, the amount of data that was sent is returned in the TPLCOUNT field of the TPL.

Note OPTCD=BLOCK | NOBLOCK is a new parameter for TSEND and applies only to MODE=SOCKETS endpoints.

  - OPTCD=BLOCK: All of the data in the send request is sent.
  - OPTCD=NOBLOCK: All, some, or none of the data may be sent. The amount of data that is actually sent depends upon the space available in the current send buffer.
- Data contained in the application program's storage is moved into the API address space when it is scheduled for transmission. Therefore, if the TSEND macro instruction is issued in asynchronous mode, the application may NOT reuse the storage area until the TSEND request has completed.

TSENDTO

Send a Datagram

The TSENDTO macro instruction is used to send datagrams to a remote transport user through an endpoint operating in connectionless-mode. The user data, the remote protocol address of the destination, and any options associated with the datagram are provided by the application program.

[*symbol*] TSENDTO [EP = *endpoint_id*]
[,ADLEN = *protocol_address_length*]
[,ADBUF = *protocol_address_address*]
[,ADALET = *protocol_address_alet*]
[,DALEN = *user_data_length*]
[,DABUF = *user_data_address*]
[,DAALET = *user_data_alet*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,DIRECT | INDIR]
[,MBUF | NOMBUF]
[,BLOCK | NOBLOCK])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E,[*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TSENDTO macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

ADLEN = *protocol_address_length*

Indicates the length (in bytes) of the protocol address contained in the storage area identified by the ADBUF operand. A length of zero is invalid, and causes the request to be abnormally completed.

Default: 0 (no protocol address)

ADBUF = *protocol_address_address*

Indicates the address of a storage area containing the protocol address of the destination transport user that is to receive the datagram. The length of the protocol address is designated by the ADLEN operand. The protocol address can be aligned on any boundary convenient for the application program.

Default: 0 (no protocol address storage area)

ADALET = *protocol_address_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the ADBUF parameter. The ADALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The ADALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

DALEN = *user_data_length*

Indicates the length (in bytes) of a data storage area or an indirect data vector identified by the DABUF operand.

If the data mode is direct, the amount of user data to be sent is equal to the length of the storage area.

If the data mode is indirect, the total amount of user data is equal to the sum of all data segments identified by the data vector.

In either case, the total amount of user data must not exceed the limit supported by the transport provider. This limit can be obtained with the TINFO macro instruction. A length of zero indicates there is no user data to be sent.

Default: 0 (no user data)

DABUF = *user_data_address*

Indicates the address of user data to be sent to the specified transport user.

If the data mode is direct, the value specified is the address of the storage area containing the user data.

If the data mode is indirect, the value specified must be the address of an indirect data vector, and each element of the vector must have been initialized to point to an individual segment of user data.

If no data is available, the length as indicated by the DALEN operand should be zero. The content of all user data is application-dependent, and is not interpreted by the API or the transport provider. The storage area can be aligned on any boundary convenient for the application program.

Default: 0 (no user data storage area)

DAALET = *user_data_alet*

Indicates an Access List Entry Token (ALET) that is used in access register (AR) mode when referencing the storage specified by the DABUF parameter. The DAALET value must be an ALET that is contained in the Dispatchable Unit Access List (DUAL) of the caller. The DAALET parameter may be used only if OPTCD=EXTEND is also specified.

Default: 0 (the storage is contained in the address space of the caller.)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TSENDTO macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TSENDTO request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = DIRECT | INDIR

Indicates the format of the user data parameter.

If the option is OPTCD=DIRECT, the DABUF and DALEN operands identify a storage area into which data should be received directly.

If the option is OPTCD=INDIR, the storage area identified by these operands contains an indirect data vector. An indirect data vector consists of a list of address-length pairs, with each element identifying a separate segment of non-contiguous storage. In this case, DABUF is the address of the first element in the list, and DALEN is the total length of the list. The length of the vector must be a multiple of eight, and the total amount of data that can be received is the sum of the lengths of each data segment.

Default: DIRECT (send directly from data area)

OPTCD = MBUF | NOMBUF

Indicates that the DABUF parameter is the address of a Cisco IOS for S/390 MBUF, rather than a data buffer or indirect buffer list.

If the option is OPTCD=MBUF, the DALEN parameter is ignored and the length of the data is determined from fields within the MBUF structure.

If the option is OPTCD=NOMBUF, the DABUF parameter is processed normally.

Note: OPTCD=MBUF is intended only for internal Interlink applications and is used to improve performance.

Default: NOMBUF

OPTCD = BLOCK | NOBLOCK

OPTCD=NOBLOCK may be used with endpoints opened with MODE=SOCKETS.

This option is ignored for TLI-mode endpoints, which always block until the data is passed to the local network. Normally, socket-mode endpoints do not block. However, if the amount of available buffer space is exceeded, the TSENDTO request will block by default until buffer space becomes available. OPTCD=NOBLOCK may be used in this case to prevent the endpoint from becoming blocked.

Buffer space is limited by configuration parameters and TOPTION negotiation.

Default: BLOCK

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TSENDTO macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TSENDTO macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TSENDTO macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TSENDTO macro instruction completes normally (or conditionally) when the datagram has been moved from the application program's storage area, and has been forwarded to the transport provider for sending to the destination transport user.

Normal completion of the TSENDTO macro instruction implies nothing in regard to when the datagram is actually sent, and should only be interpreted to mean that the transport provider has taken custody of the user data, and the storage area provided by the application program can be reused by another TSENDTO macro instruction.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY), and a conditional completion code is returned in register 0. The TPL return code field is set accordingly. No other information is returned.

If the TSENDTO macro instruction completes abnormally, the datagram is not sent to the destination transport user. The state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TSENDTO return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-48 TSENDTO Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY		
TRFAILED	TAEXCPTN	TENONEGO		
	TAINTEG	TEPROTO		
	TAENVIRO	TESYSERR TESTOP TEUNSUPF	TESUBSYS TETERM	TEDRAIN TEUNSUPO
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD TEBDDATA	TEBDECB TEBDOPTN
	TAPROCED	TEAMODE TEBUFOVR	TESTATE	TEREQOVR
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TSENDTO macro sends a datagram through an endpoint operating in connectionless mode. In addition to user data comprising the datagram, the application program provides the protocol address of the destination transport user and protocol options associated with the datagram. The datagram is transmitted as a single, contiguous unit of data and must be provided to the transport provider in its entirety.

The datagram may be provided as a simple, contiguous segment of storage, or as a set of non-contiguous segments indirectly addressed via a data vector.

- If the option is OPTCD=DIRECT, a complete datagram must be contained in the storage area identified by the DABUF and DALEN operands.
- If the option is OPTCD=INDIR, DABUF and DALEN identify a storage area initialized with the addresses and lengths of non-contiguous storage segments containing the datagram. The total amount of data to be transferred is the sum of the lengths of the individual segments. The total length must not exceed the maximum size of the interface data unit supported by the transport provider, or the maximum size of a transport service data unit. Upon completion of the TSENDTO request, the length of the indirect data vector is updated to reflect the actual amount of data transferred.

Each entry in an indirect data vector consists of a fullword address followed by a fullword length. If the length is zero, the entry is ignored; if the length is non-zero, the address must reference a valid storage area containing user data, and may be aligned on any boundary convenient for the application program. The length of the vector determines the number of entries in the list.

Unlike most other macro instructions, multiple TSENDTO macro instructions can be issued without waiting for the first to complete. However, each macro requires its own TPL. The maximum number that can be issued before one must complete is an API variable that can be modified by the TOPTION macro. The default value is set when the API is installed. TSENDTO macros are completed in the order in which they are issued.

Datagrams sent with the TSENDTO macro are buffered in the API address space before they are forwarded to the transport provider. The total amount of send buffering allocated for an endpoint is also an the API option.

Data contained in the application program's storage area is moved into the API address space when the TSENDTO macro is accepted. Therefore, if the TSENDTO macro instruction is executed in asynchronous mode, the application program can reuse the storage area before the macro instruction completes. In fact, the storage area may be used to send another datagram as long as an inactive TPL is available. The content of a datagram is application-dependent, and is not interpreted by the API or the transport provider. The maximum amount that can be sent with a single TSENDTO request can be determined by issuing a TINFO macro instruction.

TSTATE

Test TPL and Return Endpoint State

The TSTATE macro instruction is used to acquire the current state of an endpoint. Since the TSTATE macro instruction is TPL-based like most other API macro instructions but does not modify any fields in the TPL, it can also be used to determine the active or inactive state of a TPL.

[*symbol*] TSTATE MF = (E, *tpl_address*)

Syntax Description

MF = (E, tpl_address) — Indicates the execute form of the TSTATE macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL identifying the entry point.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: none (must be coded as indicated)

Completion Information

If the TSTATE macro instruction completes normally, the general return code is set to 0 (TROKAY), and a fullword of state information is returned in register 0. The TPL return code field is not modified, and no other information is returned.

If the TSTATE macro instruction completes abnormally, the general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. If the general return code is TRFAILED, the TPL was active, and the recovery action code indicates whether or not the active request has been posted complete. Otherwise, a fatal error occurred. The TPL return code field is not updated, and the state of the endpoint is unchanged.

Note The SYNAD or LERAD exit routines are not entered when the TSTATE macro instruction completes abnormally.

Return Codes

This table lists the symbolic names for the TSTATE return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-49 TSTATE Return Codes

Register 15	Register 0	Explanation
TROKAY	Endpoint State	The TPL is inactive, and the TPLEPID field designates a valid endpoint. The state of the endpoint is returned in register 0.
TRFAILED	TAEXCPTN	The TPL is active, and the requested operation has been posted complete. A TCHECK macro instruction does not suspend the issuing task.
TRFAILED	TATPLERR	The TPL is active, and the requested operation has not been posted complete. A TCHECK macro instruction may cause the issuing task to be suspended.
TRFATLFC	func. code	The function code loaded into register 0 was invalid.
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.

Table 1-49 TSTATE Return Codes (Continued)

Register 15	Register 0	Explanation
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.

Note Conditional or specific error codes are not applicable to the TSTATE macro.

Usage Information

The TSTATE macro instruction is used to determine the state of an endpoint, and implicitly, the state of a TPL that references it.

The TSTATE macro instruction is TPL-based, and therefore requires a TPL for making the request. If the TPL is valid and inactive, a fullword of state information is returned to the application program in register 0. The TPL is unchanged. The information returned is called a Transport Endpoint State Word (TSW), and is mapped by the TSW dsect.

The state word consists of two components: a halfword containing status bits representing pending activity on the endpoint, and a halfword state value that represents the current state of the endpoint. This state information is standard for all endpoints, and all transport providers. However, not all states are valid for a particular provider. For example, if a transport provider does not support orderly release of a connection, the endpoint can never acquire the release-in-progress state.

Table 1-50 Standard API Endpoint States (Defined in the TPL dsect

Endpoint	State	State Description
TSCLOSED	Closed	Endpoint TSCLOSED; the state of an endpoint before it is opened with a TOPEN macro instruction. By definition, an endpoint that exits cannot be in the closed state. Any the API macro instruction executed at an endpoint in the closed state results in a fatal error. An endpoint returns to the closed state when it is closed by a TCLOSE macro instruction.
TSOPENED	Opened	Endpoint TSOPENED; the state of an endpoint immediately after being opened with a TOPEN macro instruction. An endpoint in the opened state is not associated with any local protocol address, and cannot receive inbound or outbound connection requests. An endpoint returns to the opened state after being unbound with a TUNBIND macro instruction.
TSDSABLD	Disabled	Endpoint TSDSABLD; the state of an endpoint immediately after a local protocol address has been bound with a TBIND macro instruction, and before it is enabled to receive connect indications. An endpoint in the disabled state and operating in connectionless mode is ready to send or receive datagrams. An endpoint in the disabled state and operating in connection mode is ready to initiate a connection request. A client-mode endpoint returns to the disabled state after a connection has been released.

Table 1-50 Standard API Endpoint States (Defined in the TPL dsect (Continued)

TSINCONN	Enabled	Endpoint TSENABLD; the state of an endpoint after a local protocol address has been bound, and a non-zero value for QLSTN is specified in the TBIND macro instruction. An endpoint in the enabled state cannot operate in connectionless mode. An endpoint in the enabled state and operating in connection mode is ready to receive connect indications. A server-mode endpoint returns to the enabled state after a connect indication is accepted (multi-threaded mode) or rejected, or when the connection is released.
TSINCONN	Connect-indication-pending	Endpoint TSINCONN; the state of an endpoint when one or more connect indications have been received with the TLISTEN macro instruction that have not been accepted or rejected by the application program. The endpoint remains in the connect-indication-pending state as long as at least one indication remains pending, even though some have been accepted or rejected.
TSOUCONN	Connection-in-progress	Endpoint TSOUCONN; the state of an endpoint when a TCONNECT macro instruction has been executed, and a connect confirmation has not been received by the application program. The endpoint remains in the connect-in-progress state until a TCONFIRM macro instruction is executed to receive the connect confirmation.
TSCONNCT	Connected	Endpoint TSCONNCT; the state of an endpoint after a connect indication has been accepted with a TACCEPT macro instruction, or a confirm indication has been received with a TCONFIRM macro instruction. In single-threaded mode, the endpoint that received the connect indication enters the connected state; in multi-threaded mode, the connection is accepted to a disabled endpoint, causing it to enter the connected state. An endpoint in the connected state and operating in connection mode is ready to send and receive data. An endpoint operating in connectionless mode enters the connected state when an association has been established, and also becomes ready to send and receive datagrams with the TSEND and TRECV macro instructions.
TSINRLSE	Release-indication -pending	Endpoint TSINRLSE; the state of an endpoint that was connected after a TRELACK macro instruction is executed to acknowledge an orderly release indication. The application program may continue sending data through the endpoint. The endpoint remains in the release-indication-pending state until a TRELEASE macro instruction is executed, at which time it returns to the enabled or disabled state.
TSOURLSE	Release-in-progress	Endpoint TSOURLSE; the state of an endpoint that was connected after a TRELEASE macro instruction is executed. The application program may continue receiving data arriving at the endpoint. The endpoint remains in the release-in-progress state until a release indication is acknowledged with the TRELACK macro instruction, at which time it returns to the enabled or disabled state.

State transitions occur when the macro instruction that causes the transition completes normally and the request is posted complete, either by posting the ECB associated with the TPL, or entering the TPL exit routine. A state transition never occurs when a macro instruction completes abnormally.

If the TPL is active, the TSTATE macro instruction completes abnormally. The recovery action code returned in register 0 can be tested to determine if the active request has been posted complete. If it has, a TCHECK macro instruction can be executed at the TPL without causing a system WAIT to be issued by the API. Thus, the TSTATE macro instruction can be used to poll the TPL to determine when it is safe to issue a TCHECK macro instruction without suspending the issuing task.

A TSTATE macro instruction can be executed at any endpoint using any TPL (active or inactive) without affecting the state of the endpoint, or modifying the TPL.

TUNBIND

Unbind Protocol Address from Endpoint

The TUNBIND macro instruction is used to disable an endpoint and unbind the local protocol address that was previously bound to it with a TBIND macro instruction. Once disabled, the endpoint can no longer receive connect indications, or be used to initiate a connection.

[*symbol*] TUNBIND [EP = *endpoint_id*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E,[*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TUNBIND macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TUNBIND macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TUNBIND request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TUNBIND macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TUNBIND macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TUNBIND macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TUNBIND macro instruction completes normally when the local protocol address has been unbound, and the endpoint has been disabled. The state of the endpoint is changed from disabled (TSDSABLD) or enabled (TSENABLD) to opened (TSOPENED).

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY). The conditional completion code in register 0 is always 0 (TCOKAY), and the TPL return code field is set accordingly. No other information is returned.

If the TUNBIND macro instruction completes abnormally, the state of the endpoint remains unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TUNBIND return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-51 TUNBIND Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY		
	TAENVIRO	TESYSERR TESTOP	TESUBSYS TETERM	TEDRAIN
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD	TEBDECB
	TAPROCED	TEAMODE	TESTATE	TEINCMPL
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TUNBIND macro instruction is used to disable an endpoint and disassociate the local protocol address that was bound with the TBIND macro instruction. If the endpoint was operating in connection mode, no connection requests can be initiated or received through the endpoint. If the endpoint was operating in connectionless mode, no datagrams can be sent or received through the endpoint.

After a TUNBIND macro instruction has been executed, new options can be specified, and another local protocol address can be bound to the endpoint.

TUSER

Associate User with Endpoint

The TUSER macro instruction is used to associate a user ID with an endpoint for accounting and authorization purposes. Any SMF records written contain the user ID, and access to privileged resources or facilities is controlled by access privileges associated with the user ID, and obtained from the local security system.

[*symbol*] TUSER [EP = *endpoint_id*]
[,USER = *endpoint_userid*]
[,OPTCD = ([SHORT | LONG | EXTEND]
[,SYNC | ASYNC]
[,TUB | ACEE]
[,PLAIN | CIPHER])]
[,ECB = INTERNAL | *event_control_block_addr*]
[,EXIT = *tpl_exit_routine_address*]
[,MF = (I | L | G | M | E,[*tpl_address*])]

Syntax Description

EP = *endpoint_id*

Specifies the endpoint at which the TUSER macro instruction is to be executed. The value specified must be the endpoint identifier returned by the TOPEN macro instruction when the endpoint was opened. An invalid or corrupted value causes unpredictable results.

Default: 0 (no endpoint specified)

USER =*endpoint_userid*

Associates a user ID with the endpoint for authorization and accounting purposes.

If the option is OPTCD=TUB, the specified value must be the address of a Transport Endpoint User Block (TUB) containing the user information.

If the option is OPTCD=ACEE, the specified value must be the address of an Accessor Environment Element (ACEE) obtained from the local security system when the user ID was authenticated.

If this operand is not coded, the application name specified in the APCB is used.

The password contained in the TUB may be plain text or cipher text depending on the OPTCD=PLAIN | CIPHER operand. If cipher text, it is assumed that the password was encrypted using the encryption mechanism supplied by the local security system. The API merely provides the password to the security system in its encrypted form.

The user ID or application name is also supplied to the transport provider. How this information is used is unspecified, and provider-dependent.

Default: 0 (no user ID; use application name for accounting and authorization)

OPTCD = SHORT | LONG | EXTEND

Indicates the format attribute of the parameter list associated with this request.

If the option is OPTCD=SHORT, a different control block identifier is used to indicate that a subset of the TPL has been generated.

If the option is OPTCD=LONG, a standard, full-length TPL is generated.

If the option is OPTCD=EXTEND, an additional suffix to the standard length TPL is generated. The suffix is used to contain ALET address extensions that may be specified by other request parameters.

Default: SHORT if MF=I or MF operand omitted, LONG otherwise

OPTCD = SYNC | ASYNC

Indicates the synchronization mode to be used when executing the TUSER macro instruction.

If the option is OPTCD=SYNC, the request is executed in synchronous mode, and control is not returned to the application program until the requested macro instruction is complete. A TCHECK macro instruction should not be executed since check processing is automatically performed by the API.

If the option is OPTCD=ASYNC, the request is executed in asynchronous mode, and control is returned immediately after scheduling the TUSER request. The application program is responsible for issuing the TCHECK macro instruction.

Default: SYNC (synchronous mode)

OPTCD = TUB | ACEE

Indicates the format of user ID information referenced by the USER operand.

If the option is OPTCD=TUB, user ID, group, and password information is provided in a Transport User Block (TUB).

If the option is OPTCD=ACEE, the user information is contained in an Accessor Environment Element (ACEE) obtained from the local security system.

Default: TUB (user information provided in TUB)

OPTCD = PLAIN | CIPHER

Indicates whether the password contained in the Transport User Block (TUB) designated with the USER operand has been encrypted, or is in its plain text form.

If the option is OPTCD=PLAIN, the password is in plain text.

If the option is OPTCD=CIPHER, the password is encrypted.

The API uses this information when requesting user ID and password verification from the local security system.

Default: PLAIN (password in plain text)

ECB = INTERNAL | *event_control_block_addr*

Indicates the location of an Event Control Block (ECB) to be posted by the API when the TUSER macro instruction associated with this TPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary. If ECB=INTERNAL is coded, the TPL field normally used to store the ECB address is used as an internal ECB.

The ECB operand should only be coded when asynchronous mode is specified. In synchronous mode, the request is treated as if ECB=INTERNAL was coded, and any value specified with the ECB operand is overwritten by the internal ECB.

This operand is mutually exclusive with the following EXIT operand.

Default: INTERNAL (internal ECB)

EXIT = *tpl_exit_routine_address*

Indicates the address of an exit routine to be scheduled when the TUSER macro instruction associated with this TPL is completed. The TPL exit routine is scheduled only if asynchronous mode has been specified. In synchronous mode, any address specified with the EXIT operand is overwritten by an internal ECB.

This operand is mutually exclusive with the previous ECB operand.

Default: not indicated (no TPL exit routine)

MF = (I | L | G | M | E, [*tpl_address*])

Indicates the standard, list, generate, modify, or execute form of the TUSER macro instruction. The second sublist operand, tpl_address, specifies the address of the TPL to be used for this request. If no MF operand is specified, the standard form is used.

Read List, Generate, Modify, and Execute Forms for valid combinations of the MF subparameters.

Default: MF=I (standard, nonreentrant form)

Completion Information

The TUSER macro instruction completes normally when the designated user has been associated with the endpoint. All subsequent macro instructions issued at the endpoint are executed with the access privileges of the new user. An SMF record may be written to account for resources utilized by the previous user.

Note Currently, no authorization checking is performed.

On normal return to the application program, the general return code in register 15 is set to 0 (TROKAY). The conditional completion code in register 0 is always 0 (TCOKAY), and the TPL return code field is set accordingly. No other information is returned.

If the TUSER macro instruction completes abnormally, the previous user continues to be associated with the endpoint, and the state of the endpoint is unchanged. The general return code in register 15, and recovery action code in register 0, indicate the nature of the failure. The TPL return code field may also contain a specific error code that identifies a particular error.

Return Codes

This table lists the symbolic names for the TUSER return codes. The values associated with the symbolic names can be found in the TPL macro expansion.

Table 1-52 TUSER Return Codes

General Return Code (Register 15)	Recovery Action Code (Register 0)	Conditional Or Specific Error Code/Explanation		
TROKAY	TAOKAY	TCOKAY		
TRFAILED	TAENVIRO	TESYSERR TESTOP	TESUBSYS TETERM	TEDRAIN TEUNAUTH
	TAFORMAT	TEBDFNCD TEBDEXIT	TEBDOPCD TEBDUSER	TEBDECB TEBDACEE
	TAPROCED	TEAMODE	TESTATE	TEINCMPL
	TATPLERR	TEACTIVE		
TRFATLFC	func. code	The function code loaded into register 0 was invalid.		
TRFATLPL	diag. code	The TPL address was invalid, or the TPL has been corrupted.		
TRFATLAM	diag. code	A fatal access method error occurred, most likely due to corrupted data areas maintained within the application program's address space.		
TRFATLAP	diag. code	The APCB associated with the transport user has been closed, or is in the process of closing.		

Usage Information

The TUSER macro instruction is used to associate a new user with an endpoint for authorization and accounting purposes. The new user is identified by a Transport User Block (TUB) or an Accessor Environment Element (ACEE). The access privileges of the new user replace those of the user (if any) defined when the endpoint was opened with a TOPEN macro instruction.

If user information is provided with a TUB, the API authenticates the user ID, group name, and password combination using the local security system. Otherwise, the application must authenticate the user, and provide the address of an ACEE created by the security system.

The TUSER macro instruction is provided for multiple-user application programs that implement a logon procedure. Since the logon information must be obtained from the user after a connection has been established, the user ID to be associated with the endpoint is not known when the endpoint is created. In this case, the TOPEN macro instruction should associate the endpoint with the used ID of an overhead account, or use the default privileges associated with the application name (see APCB). When the logon procedure has been completed, the real user can be associated with the endpoint by issuing a TUSER macro instruction.