Information Management for z/OS

Terminal Simulator Guide and Reference

Version 7.1

SC31-8755-00

Information Management for z/OS

Terminal Simulator Guide and Reference

Version 7.1

SC31-8755-00

Tivoli Information Management for z/OS Terminal Simulator Guide and Reference

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both: IBM, the IBM logo, Tivoli, the Tivoli logo, AIX, CICS, CICS/ESA, DATABASE 2, DB2, DFSMS/MVS, IBMLink, Language Environment, MVS, MVS/ESA, NetView, OS/2, OS/2 WARP, OS/390, RACF, Redbooks, RMF, System/390, Tivoli Enterprise Console, TME 10, VTAM, z/OS.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names mentioned in this document may be trademarks or service marks of others.

Notices

References in this publication to Tivoli Systems or IBM products, programs, or services do not imply that they will be available in all countries in which Tivoli Systems or IBM operates. Any reference to these products, programs, or services is not intended to imply that only Tivoli Systems or IBM products, programs, or services can be used. Subject to valid intellectual property or other legally protectable right of Tivoli Systems or IBM, any functionally equivalent product, program, or service can be used instead of the referenced product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by Tivoli Systems or IBM, are the responsibility of the user. Tivoli Systems or IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, New York 10504-1785, U.S.A.

Programming Interface Information

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of Tivoli Information Management for z/OS.

Contents

Preface

The Terminal Simulator Facility of Tivoli® Information Management for z/OS enables you to control input and output operations from either a batch or interactive environment. You can design and build as many TSPs or TSXs as you want to meet needs unique to your installation. You build your TSPs by using the Panel Modification Facility of Tivoli Information Management for z/OS to create special control lines. You code your TSXs using REXX.

A TSP or TSX can simulate an entire interactive session between you and the Tivoli Information Management for z/OS product. Because it can run in a batch environment, a TSP or TSX can do a great deal of work during off-shift hours, without the need for direct user interaction with Tivoli Information Management for z/OS.

This is not only a guide to creating and using TSPs and TSXs, but it is also a reference for control line functions. It gives you step-by-step instructions for designing, building, and testing a sample TSP. These instructions are followed by information about the different ways you can run TSPs and TSXs.

This publication describes Tivoli Information Management for z/OS Version 7 Release 1. There may be references in this publication to versions of Tivoli Information Management for z/OS's predecessor products. For example:

- TME 10™ Information/Management Version 1.1
- Information/Management Version 6.3, Version 6.2, Version 6.1
- Tivoli Service Desk for OS/390® Version 1.2

Who Should Read This Guide

This guide is intended for users who are experienced in tailoring the Tivoli Information Management for z/OS product to the needs of a data processing installation.

To use this guide effectively, you must understand structured words (s-words), prefix words (p-words), and the format and structure of your Tivoli Information Management for z/OS databases. You can find information about s-words and p-words in the *Tivoli Information Management for z/OS Panel Modification Facility Guide*. Information about database format and structure is in the *Tivoli Information Management for z/OS Planning and Installation Guide and Reference*. In order to write TSPs, you must also be familiar with the Tivoli Information Management for z/OS Panel Modification Facility (PMF). In order to write TSXs, you must be familiar with REXX. You also need to understand the sequence in which interactive panels are displayed (the panel flow) for your particular Tivoli Information Management for z/OS installation. Information about PMF can be found in the *Tivoli Information Management for z/OS Panel Modification Facility Guide*.

Prerequisite and Related Documentation

The library for Tivoli Information Management for z/OS Version 7.1 consists of these publications. For a description of each, see "The Tivoli Information Management for z/OS Library" on page 311.

Tivoli Information Management for z/OS Application Program Interface Guide, SC31-8737-00

Tivoli Information Management for z/OS Client Installation and User's Guide, SC31-8738-00

Tivoli Information Management for z/OS Data Reporting User's Guide, SC31-8739-00

Tivoli Information Management for z/OS Desktop User's Guide, SC31-8740-00

Tivoli Information Management for z/OS Diagnosis Guide, GC31-8741-00

Tivoli Information Management for z/OS Guide to Integrating with Tivoli Applications, SC31-8744-00

Tivoli Information Management for z/OS Integration Facility Guide, SC31-8745-00

Tivoli Information Management for z/OS Licensed Program Specification, GC31-8746-00

Tivoli Information Management for z/OS Master Index, Glossary, and Bibliography, SC31-8747-00

Tivoli Information Management for z/OS Messages and Codes, GC31-8748-00

Tivoli Information Management for z/OS Operation and Maintenance Reference, SC31-8749-00

Tivoli Information Management for z/OS Panel Modification Facility Guide, SC31-8750-00

Tivoli Information Management for z/OS Planning and Installation Guide and Reference, GC31-8751-00

Tivoli Information Management for z/OS Problem, Change, and Configuration Management, SC31-8752-00

Tivoli Information Management for z/OS Program Administration Guide and Reference, SC31-8753-00

Tivoli Information Management for z/OS Reference Summary, SC31-8754-00

Tivoli Information Management for z/OS Terminal Simulator Guide and Reference, SC31-8755-00

Tivoli Information Management for z/OS User's Guide , SC31-8756-00

Tivoli Information Management for z/OS World Wide Web Interface Guide, SC31-8757-00

Note: Tivoli is in the process of changing product names. Products referenced in this manual may still be available under their old names (for example, TME 10 Enterprise Console instead of Tivoli Enterprise Console®).

What This Guide Contains

This guide is divided into the following sections:

- "What Is Terminal Simulation?" on page 1 presents basic information about TSPs and TSXs.
- "Designing and Creating a Terminal Simulator Panel (TSP)" on page 5 takes you step-by-step through task analysis, TSP design, and implementation.
- "Designing and Creating a Terminal Simulator EXEC (TSX)" on page 53 takes you through the steps required to design and implement a TSX.

- "Creating Terminal Simulator Control Lines" on page 61 presents basic information about TSP and TSX control lines. The control lines are listed in alphabetical order.
- "Remote Data Resource Terminal Simulator Control Lines" on page 241 describes some TSXs that can be used with Remote Data Resources (RDRs). A Remote Data Resource is an area (or areas) in the BLX-SP containing strings of character data.
- "Testing Terminal Simulator Panels (TSPs) and EXECS (TSXs)" on page 251 suggests several methods you can use to test your TSPs and TSXs before putting them into production.
- "Running Terminal Simulator Panels (TSPs) and Terminal Simulator Execs (TSXs)" on page 257 presents basic information about invoking TSPs and TSXs.
- "User Exits" on page 263 presents information about user exits that you can use in your TSPs and TSXs.
- "Terminal Simulator Communications Fields" on page 289 contains detailed information about the fields comprising the terminal simulator communications area (TSCA), a control block defined on page 2.
- "Assembler Code User Exit Example" on page 303 contains information to get you started on writing your own user exit routines.
- At the back of this book you will find::
  - Information about other books in the Tivoli Information Management for z/OS library
  - Information about classes offered by IBM® Education
  - An index for this book.

Typeface Conventions

This guide uses several typeface conventions for special terms and actions. These conventions have the following meaning:

Bold	Entries that you must use literally, choices, or options that you select is displayed in **bold**.
Italics	Variables and values that you must provide is displayed in *italics*. New terms are also displayed in italics.
`Monospace`	Code examples appear in `monospace` font.

The panels as presented in this guide are not meant to be exact replicas of the way a panel might appear on the screen. The information on the panels is correct, but the spacing is not always exact.

Commands, such as END, CONTROL, RESUME, or DOWN, appear in all capital letters in text. Although not commands, the user responses YES and NO also appear in capital letters.

Contacting Customer Support

For support inside the United States, for this or any other Tivoli product, contact Tivoli Customer Support in one of the following ways:

- Send e-mail to **support@tivoli.com**
- Call 1-800-TIVOLI8
- Navigate our Web site at **http://www.support.tivoli.com**

For support outside the United States, refer to your Customer Support Handbook for phone numbers in your country. The Customer Support Handbook is available online at **http://www.support.tivoli.com**.

When you contact Tivoli Customer Support, be prepared to provide identification information for your company so that support personnel can assist you more readily.

The latest downloads and fixes can be obtained at **http://www.tivoli.com/infoman**

1

What Is Terminal Simulation?

Tivoli Information Management for z/OS includes two methods through which you can control the simulation of terminal input and output. The first method involves the use of a Terminal Simulator *Panel* (TSP). You use the Panel Modification Facility (PMF) to create TSPs. The second method involves the use of a Terminal Simulator *EXEC* (TSX). A TSX is a REXX EXEC, and you use an editor, such as the ISPF PDF editor, to write TSXs.

Some of the functions you can perform with TSPs and TSXs are:

- Create records
- Update a group of records that meet a particular search criteria
- Delete a group of records that meet a particular search criteria
- Automate sections of a panel flow for interactive users
- Prime fields in a record
- Cross-check values entered in multiple fields
- Issue a message
- Provide user-defined line commands
- Provide user-defined commands

Examples

The following examples show how you might use a TSP or a TSX.

Updating Records

In this example, suppose that an employee who analyzes Tivoli Information Management for z/OS problem records moves to a new office and is assigned a new phone number. The open problem records assigned to that person should be updated to reflect the employee's new phone number. You can build a TSP or a TSX that does a search of all open records that list that employee as the assignee. The TSP or TSX can sequentially update each record and change the old phone number to the new phone number. You accomplish a tedious job without tying up an interactive user or seriously affecting other users of the system.

Managing Data

In this example, suppose that you want to remove obsolete records from the Tivoli Information Management for z/OS database and retain a record of the data in another data set. To do this, you can build a TSP or a TSX that transfers specific records from a database to another storage medium, such as tape, and then delete the records from the original database. The TSP or TSX searches all records for those that match a specific criterion, such as "closed before 1998."

The TSP or TSX then processes each record from the search results list, creates a sequential form of the record, and writes the sequential record to a data set. After copying all records to the data set, the TSP or TSX sequentially deletes each record in the search results list from the original database.

Note: Refer to "FLATTEN" on page 111 for additional information on how to create a sequential form of a record.

Suppose then that you want to improve the performance of your remaining database. You can build a TSP to uncognize s-words that your users rarely use in searches. For each s-word, the TSP searches all records for those containing the target s-word. The TSP then updates each record from the search results list, changing the data that controls cognizing.

Note: Refer to "WORDFIX" on page 207 for additional information on how to update information in records. The change function of WORDFIX is not supported in a TSX, so in order to perform the change function, you will need to use the LINK control line to link to a TSP to perform WORDFIX. You can use the TSX ADDSDATA control line (described in "ADDSDATA" on page 71) to perform WORDFIX-like add function and the DELSDATA control line (described in "DELSDATA" on page 82) to perform WORDFIX-like delete functions.

TSP and TSX Control Lines

To perform these functions, Tivoli Information Management for z/OS provides a set of control lines. In a TSP, a control line is similar to a macro instruction. In a TSX, a control line is a REXX CALL statement. Most of the control lines originally supported by TSPs are now supported by TSXs. There are also some new control lines supported only by TSXs. Refer to Table 1 on page 61 for additional information about which control lines are supported by TSPs, by TSXs, or by both TSPs and TSXs.

If you want a TSP or a TSX to perform a function that cannot be done with any of the control lines, you can write a program that interfaces directly with the TSCA (see "The Terminal Simulator Communications Area" for additional information on the TSCA). You call such programs (referred to as *user exit routines* throughout this book) by means of a control line called USEREXIT. User exit routines pass data through the TSCA to Tivoli Information Management for z/OS for processing. Similarly, Tivoli Information Management for z/OS stores data in the TSCA that the user exit routines can retrieve. "USEREXIT" on page 193 discusses the USEREXIT control line.

The Terminal Simulator Communications Area

The *Terminal Simulator Communications Area* (TSCA) provides the storage for retrieval of information communicated between the Tivoli Information Management for z/OS product, TSPs and TSXs, and user-written exit routines. The TSCA is a control block that contains flag indicator fields, pointer fields, and data fields. When you start a TSP or a TSX, Tivoli Information Management for z/OS initializes the control block. As the TSP or TSX runs, data is written into or read from the TSCA. For more detailed information about the TSCA, see "Terminal Simulator Communications Fields" on page 289.

You can modify TSCA fields directly with a TSP. However, changes to some fields cause problems. The tables shown in Terminal Simulator Communications Fields, indicate the fields you must not modify with a TSP. This appendix also includes an assembler language mapping of the TSCA.

Note: In a TSX, TSCA fields can only be modified by TSX control lines. You cannot directly modify TSCA fields in a TSX.

Should You Use a TSP or a TSX?

Once you decide to use terminal simulation to perform a Tivoli Information Management for z/OS task, you must decide whether to use a TSP, a TSX, or a combination of the two. You should consider your skills. Have you ever written a REXX program? Writing a TSX is similar to writing a REXX program. Have you ever used PMF to create a Tivoli Information Management for z/OS panel, in particular a TSP? Writing a TSP is similar to writing a Tivoli Information Management for z/OS Control Panel.

You must also consider the functions provided by the TSP and TSX control lines. If you need to change existing data using WORDFIX, you must use a TSP, because while the TSX supports the add function (ADDSDATA) and delete function (DELSDATA), it does not have the change function that exists with the TSP. There are special TSX control lines which make processing list processor data, freeform text, and flattened records much easier in a TSX than in a TSP. You can perform terminal simulation using both TSPs and TSXs and use the LINK control line to flow between separate TSPs and TSXs. Refer to “LINK” on page 130 for additional information about the LINK control line.

2

Designing and Creating a Terminal Simulator Panel (TSP)

A TSP enables you to perform tasks using Tivoli Information Management for z/OS by simulating interactive control. You use control lines to write a TSP. Sometimes you use Tivoli Information Management for z/OS prefix words (p-words) and structured words (s-words) in your control lines. The *Tivoli Information Management for z/OS Panel Modification Facility Guide* contains additional information about using p-words and s-words.

This chapter describes steps you can use to design a TSP. It also shows the panel flow that you can use to create a TSP.

What Are Control Lines?

Control lines are macro-like terms that define functions to Tivoli Information Management for z/OS. You use them in the TSP to define what you want to do.

Attention: If you do not use the FLATTEN, UNFLATTEN, and WORDFIX control lines correctly, they can damage your existing databases. The *Tivoli Information Management for z/OS Planning and Installation Guide and Reference* contains additional information about how to prevent the misuse of these three control lines, and also discusses general data integrity and security issues regarding the use of TSPs.

The control lines available to you in a TSP are:

ADDDATA Simulates data responses that you enter on the command line of a panel in interactive mode.

BRANCH Changes the flow of the control lines.

CLEAR Discards collected data from the TSCA command line reply buffer.

FINDSDATA Extracts data from a record.

FINDSJRNL Extracts history data from the journal portion of a record.

FLATTEN Copies a record from a Tivoli Information Management for z/OS database.

ISPEXEC Calls ISPF dialog management services.

LABEL Identifies a line in the TSP for the purpose of branching. It can also be used to add comments to a TSP.

LINK Transfers control to another TSP.

MESSAGE Generates Tivoli Information Management for z/OS and user-written messages.

MOVEVAR Adds data to the variable data area of the TSCA.

PRINT Prints messages, panels, and the contents of the TSCA.

PROCESS Sends one or more responses to Tivoli Information Management for z/OS for processing. This control line must follow an ADDDATA to process the data response.

RETURN Exits the TSP.

SETFIELD Sets a field in the TSCA to communicate between TSPs or between a TSP and a user exit routine.

TESTFIELD Tests fields in the TSCA.

TESTFLOW Tests for a specified panel or message ID.

TRACE Traces the flow of control lines.

UNFLATTEN
Restores a record that was previously copied from a Tivoli Information Management for z/OS database using the FLATTEN control line.

USEREXIT Calls a user exit routine.

WORDFIX Repairs records by either deleting or changing the existing data. It can also add new data to existing records.

The rules for using these control lines and examples for using them are explained in "Creating Terminal Simulator Control Lines" on page 61.

Designing a Terminal Simulator Panel

Designing a TSP is similar to designing a program. You define the task, define the steps needed to perform the task, and then translate these steps into the appropriate TSP control lines. After designing the TSP, you use the Panel Modification Facility to create it. To help you understand TSPs, here is an example of the TSP design process.

Define the task:

Search all problem records for problems assigned to Smith. Reassign those problems to Jones.

Define the steps:
1. Search for all problem records containing the assignee name of Smith.
2. Request each record for updating.
3. Change the assignee name to Jones.
4. File each changed record.

Design the TSP:

Because a TSP simulates a Tivoli Information Management for z/OS session, first list how you perform these steps interactively with Tivoli Information Management for z/OS. The following steps accomplish the task defined previously:
1. Search problem records + PERA/SMITH.
2. Request each record in the search results list for updating, using the block line command UU (UPDATE).

3. Update the **Assignee name** field and file the changed record. (On the Problem Summary panel (BLG0BU00), enter **2,1,JONES,,9**)

Here is how you perform these same steps using TSP control lines:

1. To simulate the keystrokes needed to specify the search, use the ADDDATA control line. Then use the PROCESS control line to run the search.

```
ADDDATA   3,2,6,1,SE + PERA/SMITH
PROCESS   ERROR
```

2. To simulate the keystrokes needed to request a group of records in the search results list for updating, use the ADDDATA control line. Then use the PROCESS control line to open the records.

```
ADDDATA   LINECMD UU,DOWN LAST,LINECMD UU
PROCESS   ERROR
```

3. To simulate the keystrokes needed to update the assignee name field and file the record, use the ADDDATA control line. Then use the PROCESS control line to update and file each record.

```
ADDDATA   2,1,JONES,,9
PROCESS   ERROR
```

You might find it helpful to write out your TSP on paper before you use PMF to create it.

Note: Because most data entry and display operations for date fields use external date format, and because different users can use different external date formats, it is recommended that you use internal date format YYYY/MM/DD for all date processing. Use BLGIDATE to convert any date values retrieved from a record to internal format, then perform any required processing, and then use BLGEDATE to convert an internal date to external date format before entering the date in a field with the PROCESS control line. The user exits BLGIDATE and BLGEDATE are described in "General-purpose User Exits" on page 279.

Specifying the Search:

When you first create your TSP, you enter its name on the PMF Panel Name Entry panel (BLM8CU00). Tivoli Information Management for z/OS uses this entry to automatically create the first line of your TSP for you. It is a LABEL control line with the label name set to the panel name you specified on panel BLM8CU00. If you named your TSP UPDATEPR, the first line is:

```
LABEL      UPDATEPR
```

You can, however, delete this line if you have no need for it.

Add the TRACE control line to verify the flow of TSP control lines as you test your TSP:

```
TRACE
```

After your TSP is ready for production use, remove TRACE from the production TSP if you do not want users to receive the output that the TRACE control line produces. If you prefer, you can use the TRACE command instead. The TRACE command does interactively what the TRACE control line does in a TSP.

Add control lines to perform the search and to go to label ERROR if an error occurs at this control line when you run the TSP:

```
ADDDATA    3,2,6,1,SE + PERA/SMITH
PROCESS    ERROR
```

Note: ERROR is a target label to which the TSP branches for further processing if an error occurs at this point in your TSP. You must define ERROR in a LABEL control line where the TSP processing is supposed to continue.

If the search finds no records, exit the TSP. You can test for this condition with a control line to verify that message BLG19214 is issued when no records are found. The following control line tests for this message and goes to label DONE if the message is issued:

```
TESTFLOW   DONE
```

Note: DONE is a target label to which the TSP branches for further processing if an error occurs at this point in your TSP. You must define DONE in a LABEL control line where the TSP processing is supposed to continue.

On the TESTFLOW Specification panel (BLM8CU9K), you must enter the message name (**BLG19214**) in the **Verify name** field and **message** in the **Verify type** field.

If the TSP finds only one record, use the U line command instead of UU to update the single record. Add the following control lines after the TESTFLOW control line to test whether the search finds only one record and to go to label JUSTONE if this is true:

```
TESTFIELD  JUSTONE               (test if TSCATPLC=1; if yes, go to label JUSTONE)
```

Note: JUSTONE is a target label to which the TSP branches if it finds only one record. You must define JUSTONE in a LABEL control line where the TSP processing is supposed to continue. TSCATPLC is the field that contains the number of records that were found by the search.

Specifying How to Update the Records:

The next part of the task is to define the control lines that request each record for updating, change the assignee name to JONES, and file each record. After filing each record, test whether the current panel is the Problem Summary panel (BLG0BU00). If it is, go to label UPDATE to change the next record. If the current panel is not the Problem Summary panel, branch to label DONE. Add the following control lines to do this:

```
ADDDATA    LINECMD UU,DOWN LAST,LINECMD UU (access records for updating)
PROCESS    ERROR
LABEL      UPDATE
ADDDATA    2,1,JONES,,9
PROCESS    ERROR
TESTFLOW   UPDATE (test if the current panel is BLG0BU00, the Problem Summary panel)
BRANCH     DONE
```

Now add control lines to handle the case when the TESTFIELD control line finds only one record and branches to label JUSTONE. Update the single record, change the assignee name to JONES, then file the record:

```
LABEL      JUSTONE
ADDDATA    LINECMD U,2,1,JONES,,9
PROCESS    ERROR
```

Add control lines for the functions you want to perform when the TSP goes to label DONE. Because the TSP has finished running, exit the TSP:

```
LABEL      DONE
RETURN
```

Finally, add control lines for the functions you want to perform when your TSP goes to the error routine at label ERROR. Use the PRINT control line to print the TSCA, messages, and current panel. Then exit the TSP:

```
LABEL      ERROR
PRINT                      (print the TSCA, messages, and panel)
RETURN
```

Your final TSP looks like this:

```
LABEL      UPDATEPR
TRACE                                    (trace TSP flow)
ADDDATA    3,2,6,1,SE + PERA/SMITH       (problem record search)
PROCESS    ERROR
TESTFLOW   DONE                          (any records found?)
TESTFIELD  JUSTONE                       (test if only one record found)
ADDDATA    LINECMD UU,DOWN LAST,LINECMD UU (access records for updating)
PROCESS    ERROR
LABEL      UPDATE
ADDDATA    2,1,JONES,,9                  (change record)
PROCESS    ERROR
TESTFLOW   UPDATE                        (if not last record, continue)
BRANCH     DONE
LABEL      JUSTONE
ADDDATA    LINECMD U,2,1,JONES,,9        (only one record, update it)
PROCESS    ERROR
LABEL      DONE
RETURN
LABEL      ERROR
PRINT                                    (print TSCA, messages, panel)
RETURN
```

This TSP leaves you in the search results list. From here, you can either use the CANCEL or INIT command to end.

Creating a Terminal Simulator Panel Flow

When you finish designing a TSP, you create it with the Panel Modification Facility (PMF). PMF uses Tivoli Information Management for z/OS data sets called the *read panel data set* and the *write panel data set*. Whenever you file a new or updated panel in PMF, the panel is put into the write panel data set. While the panel remains in this data set, you can modify it. After you complete testing of the panel and it reaches its final, *production* form, you must copy it into a read panel data set so it can run in production. The *Tivoli Information Management for z/OS Planning and Installation Guide and Reference* contains additional information about these data sets.

The following panel flow begins with Tivoli Information Management for z/OS's Primary Options Menu for the System application (BLG0EN10). It creates a TSP named UPDATEPR, the same example used earlier in this chapter to illustrate the design process.

The information you enter is highlighted on each panel. Information that helps you understand the panel flow is to the left of the panel.

The panels used to collect control line information are not shown in this section. See "Creating Terminal Simulator Control Lines" on page 61 for more information about using the specification panels.

To create a new TSP, type **9,1** on the command line and press **Enter**.

```
BLG0EN20              --- PRIMARY OPTIONS MENU ---     APPLICATION: MANAGEMENT

OPTIONS:

      1. OVERVIEW.......Display general information and product enhancements.
      2. PROFILE........Display or alter invocation or session defaults.
      3. APPLICATION....Change application, list available applications.
      4. CLASS..........Change current class, list available classes.
      5. ENTRY..........Create a record.
      6. INQUIRY........Search for records.
      7. UTILITY........Copy, display, print, delete, and update records.
      8. GLOSSARY.......Display a list of searchable words in the database.
      9. PMF............Modify or create panels.


           Select an option, enter a command, or type QUIT to exit.



          Tivoli Information Management for z/OS Version 7 Release 1
                5697-SD9 (C) Copyright IBM Corp., 1981, 2001.



===> 9,1
```

The Panel Name Entry panel BLM8CU00 is displayed.

```
BLM8CU00                      PANEL NAME ENTRY                          UPDATE

Identify panel to be updated; cursor placement or input line entry allowed.




              1. Panel name.................<R> ________
              2. Data set definition label..... ________




   To enter the panel update dialog, press Enter without field modifications.









===>
```

To create a new panel, the name you enter in the **Panel name** field must be a unique name. The panels that Tivoli ships with this product start with the letters BLG, BLH, BLM, BLX, BTN, or EYM. *All* panels (those from Tivoli as well as those that you create) must have unique names. You can name your panels whatever you want, but avoid conflicting names by reviewing the *Tivoli Information Management for z/OS Panel Modification Facility Guide* for panel naming conventions.

For this example, the new TSP is **UPDATEPR**. Type **1,updatepr** on the command line and press **Enter** twice.

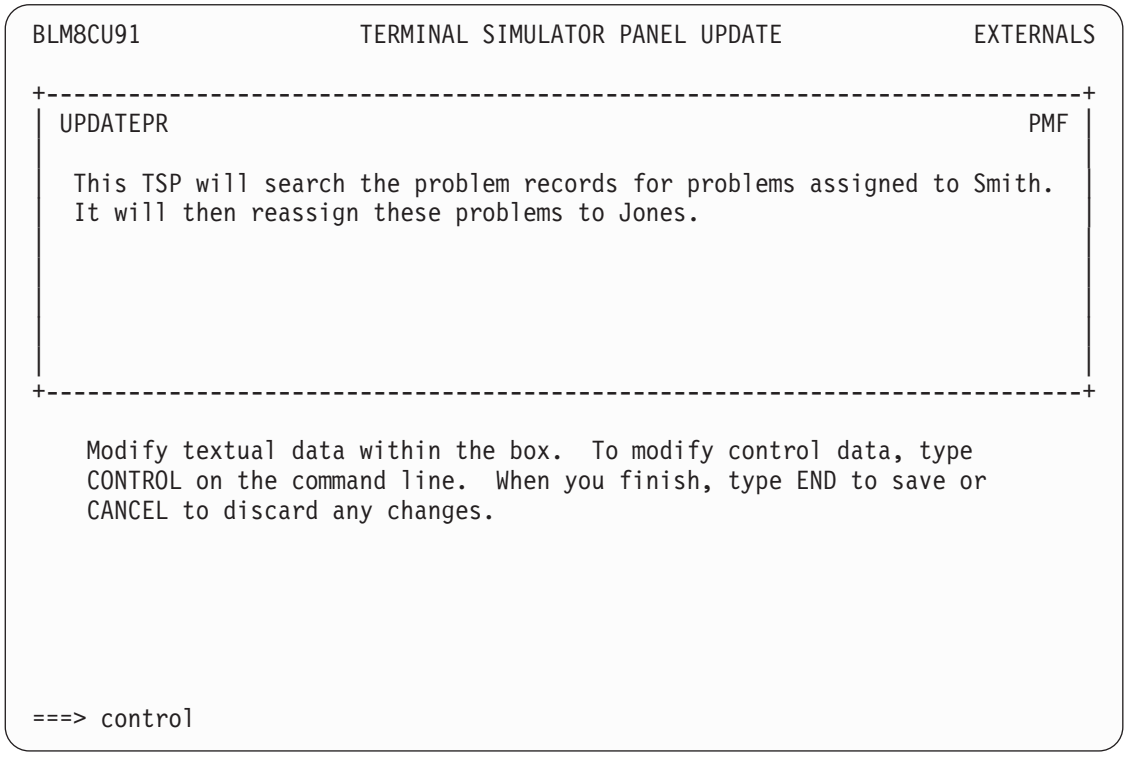
```
BLM8CU00                      PANEL NAME ENTRY                          UPDATE

Identify panel to be updated; cursor placement or input line entry allowed.




               1. Panel name.................<R> ________
               2. Data set definition label..... ________




  To enter the panel update dialog, press Enter without field modifications.









===> 1,updatepr
```

Because this TSP does not yet exist, you will receive the message BLM04030I advising you that the panel you are creating cannot be found in any panel data set. The Panel Type panel BLM8CUA0 is displayed.

To identify your panel as a TSP, type **9** and press Enter.

```
+ BLM8CUA0 ----------------- PANEL TYPE ------------------- 1 of 1-+
|                                                                   |
| USE...Identify the type of panel that you wish to have created.   |
|                                                                   |
| 1.SELECTION.........Panel containing menu selections.             |
| 2.OPTIONS...........Panel containing dialog begin options.        |
| 3.HELP..............Panel for help or tutorial information.      |
| 4.MESSAGE...........Panel containing message information.         |
| 5.DATA ENTRY.........Panel allowing display or entry fields.      |
| 6.ASSISTED ENTRY.....Panel containing value definitions.          |
| 7.TABLE.............Panel allowing multiple columns or lines.    |
| 8.CONTROL...........Panel for testing flow, program invocation.   |
| 9.TERMINAL SIMULATOR.Panel for interacting with Info/Management.  |
+------------------------ SELECT ITEM ----------------------------+




BLM04030I Panel UPDATEPR was not found in any panel data set.
===> 9
```

You can enter text that describes the purpose of this TSP. After you do that you are ready to modify the control data, so type **control** on the command line and press **Enter**.


BLM8CU91 TERMINAL SIMULATOR PANEL UPDATE EXTERNALS
UPDATEPR PMF
This TSP will search the problem records for problems assigned to Smith.
It will then reassign these problems to Jones.
Modify textual data within the box. To modify control data, type
CONTROL on the command line. When you finish, type END to save or
CANCEL to discard any changes.
===> control

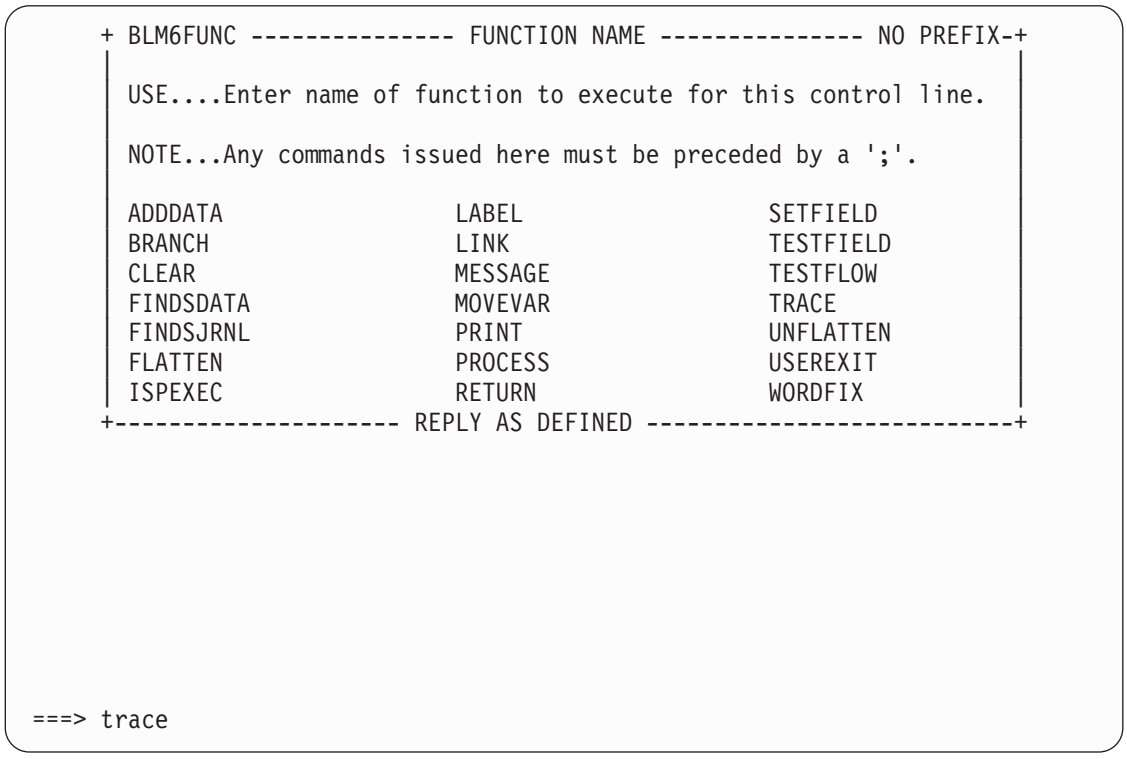
This takes you to the Function Line Summary panel BLM1TUCU, where PMF automatically creates the LABEL control line containing the name of your TSP.

To insert a new line, type **i** (INSERT) in the line command column for LABEL and press **Enter**.

```
BLM1TUCU                   FUNCTION LINE SUMMARY                    LINE 1 OF 1

   FUNCTION    LABEL   LITERAL                            GET APPLY  FIELD
     NAME      NAME    DATA                               VAR  NOT   NAME

 0. ********* ******** ********************************** **** **** ********
i1. LABEL     UPDATEPR
*** ********* ******** ********************************** **** **** ********









 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===>
```

The INSERT line command causes the Function Line Summary panel (BLM6FUNC) to appear. (If you needed to, you could also have used any of the line commands (UPDATE, DELETE, COPY, MOVE, AFTER, or REPEAT on panel BLM1TUCU to modify control information.)

Specify the next control line. For this example, type **trace** and press **Enter**. (You can skip this panel by entering the control line's name on the command line of the previous panel when you use the **i** (INSERT) line command.)



The Trace Specification panel (BLM8CU9N) is displayed. The responses shown are default responses. You can change them by typing changes into the response fields and pressing **Enter**. In this sample, accept the defaults for this panel by typing **end** and press **Enter**. You will return to panel BLM1TUCU.

```
BLM8CU9N                TRACE SPECIFICATION                  PANEL: UPDATEPR

Enter 'TRACE' control data; cursor placement or input line entry allowed.



                    1. Set TRACE on.......... YES
                    2. Trace LINK function... NO_









   When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Use the **i** (INSERT) line command to add the next TSP control line, ADDDATA. To save time, type **ADDDATA** on the command line and press Enter.

```
BLM1TUCU                  FUNCTION LINE SUMMARY                   LINE 1 OF 2

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

 0. ********* ******** ******************************** **** **** ********
 1. LABEL     UPDATEPR
i2. TRACE
*** ********* ******** ******************************** **** **** ********









 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> adddata
```

The ADDDATA control line puts keystroke information into the TSCA command line buffer.

On the ADDDATA SPECIFICATION panel (BLM8CU9A), you enter the keystroke information into the **Literal data** field. Because the product interprets this keystroke information as an immediate response chain (IRC), you cannot enter this information directly from the command line. You can move the cursor to the **Literal data** field and type the information, or type **3** on the command line and enter the information on the assisted-entry panel.

For this example, enter the information directly on the **Literal data** field.

To specify the search, type:

```
3,2,6,1,se + pera/smith
```

in the **Literal data** field and press **Enter**.

Type **end** and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9A                ADDDATA SPECIFICATION                PANEL: UPDATEPR

Enter 'ADDDATA' control data; cursor placement or input line entry allowed.



 1. Structured word index.... ____            Structured word.. __________
 2. Get variable data........ NO_             Word acronym..... ______________

 3. Literal data.............. 3,2,6,1,SE + PERA/SMITH______



     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **PROCESS**.

```
BLM1TUCU                   FUNCTION LINE SUMMARY                    LINE 1 OF 3

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

 0. ********* ******** ******************************** **** **** ********
 1. LABEL     UPDATEPR
 2. TRACE
i3. ADDDATA            3,2,6,1,SE + PERA/SMITH          NO
*** ********* ******** ******************************** **** **** ********









 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> process
```

The PROCESS line is like an Enter key for the ADDDATA control line. It starts the processing of data in the TSCA command line reply buffer, in this case, data added by the preceding ADDDATA line.

The PROCESS Specification panel (BLM8CU9H) is displayed. The **Error label name** field identifies the name of the label to which your TSP goes if an error occurs while processing this control line.

For this example, type **1,error** on the command line and press **Enter**. Then, type **end** and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9H                 PROCESS SPECIFICATION                PANEL: UPDATEPR

Enter 'PROCESS' control data; cursor placement or input line entry allowed.



                1. Error label name......<R> ERROR___
                2. Save existing messages?.. NO_




     When you finish, type END to save or CANCEL to discard any changes.







===> end
```

Add the next TSP control line, **TESTFLOW**.

```
BLM1TUCU                    FUNCTION LINE SUMMARY                    LINE 1 OF 4

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

 0. ********* ******** ******************************** **** **** ********
 1. LABEL     UPDATEPR
 2. TRACE
 3. ADDDATA            3,2,6,1,SE + PERA/SMITH          NO
i4. PROCESS   ERROR
*** ********* ******** ******************************** **** **** ********









 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> testflow
```

The TESTFLOW Specification panel (BLM8CU9K) is displayed. To test for message BLG19214, type:

```
1,blg19214,2,message,3,done
```

on the command line and press **Enter**. Then, type **end** and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9K                 TESTFLOW SPECIFICATION              PANEL: UPDATEPR

Enter 'TESTFLOW' control data; cursor placement or input line entry allowed.



                   1. Verify name........... BLG19214
                   2. Verify type........<R> MESSAGE
                   3. True label.........<R> DONE
                   4. Get variable data..... NO_
                   5. Apply not logic....... NO_



     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **TESTFIELD**.

```
BLM1TUCU                    FUNCTION LINE SUMMARY                    LINE 1 OF 5

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

 0. ********* ******** ******************************** **** **** ********
 1. LABEL     UPDATEPR
 2. TRACE
 3. ADDDATA           3,2,6,1,SE + PERA/SMITH          NO
 4. PROCESS   ERROR
i5. TESTFLOW  DONE                                     NO   NO
*** ********* ******** ******************************** **** **** ********









 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> testfield
```

The TESTFIELD Specification panel (BLM8CU9J) is displayed. To test for only one record found, type:

```
1,tscatplc,4,justone,10,1
```

on the command line and press **Enter**. Then, type **end** and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9J                TESTFIELD SPECIFICATION                PANEL: UPDATEPR

Enter 'TESTFIELD' control data; cursor placement or input line entry allowed.



 1. TSCA field name....<R> TSCATPLC
    2. Get list index?.... ___
    3. List index......... 0000
 4. True label.........<R> JUSTONE_
 5. Get variable data..... NO_
 6. Find string anywhere.. NO_
 7. Find exact string..... NO_
 8. Apply not logic....... NO_
 9. Case-sensitive........ NO
10. Test data............. 1_____________________________


     When you finish, type END to save or CANCEL to discard any changes.




===> end
```

Add the next TSP control line, **ADDDATA**.

```
BLM1TUCU                    FUNCTION LINE SUMMARY                    LINE 1 OF 6

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

 0. ********* ******** ******************************** **** **** ********
 1. LABEL     UPDATEPR
 2. TRACE
 3. ADDDATA            3,2,6,1,SE + PERA/SMITH          NO
 4. PROCESS   ERROR
 5. TESTFLOW  DONE                                     NO   NO
i6. TESTFIELD JUSTONE  1                               NO   NO   TSCATPLC
*** ********* ******** ******************************** **** **** ********




 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> adddata
```

On panel BLM8CU9A, type

```
linecmd uu,down last,linecmd uu
```

in the **Literal data** field.

Then, type **end** on the command line and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9A                ADDDATA SPECIFICATION                PANEL: UPDATEPR

Enter 'ADDDATA' control data; cursor placement or input line entry allowed.


 1. Structured word index.... ____            Structured word.. __________
 2. Get variable data........ NO_             Word acronym..... ______________

 3. Literal data.............. LINECMD UU,DOWN LAST,LINECMD UU_



     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **PROCESS**.

```
BLM1TUCU                    FUNCTION LINE SUMMARY                    LINE 1 OF 7

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

 0. ********* ******** ******************************** **** **** ********
 1. LABEL     UPDATEPR
 2. TRACE
 3. ADDDATA            3,2,6,1,SE + PERA/SMITH          NO
 4. PROCESS   ERROR
 5. TESTFLOW  DONE                                     NO   NO
 6. TESTFIELD JUSTONE  1                                NO   NO   TSCATPLC
i7. ADDDATA            LINECMD UU,DOWN LAST,LINECMD UU  NO
*** ********* ******** ******************************** **** **** ********




 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> process
```

On panel BLM8CU9H, type **1,error** on the command line and press **Enter**. Then, type **end** and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9H                 PROCESS SPECIFICATION                PANEL: UPDATEPR

Enter 'PROCESS' control data; cursor placement or input line entry allowed.



                1. Error label name......<R> ERROR___
                2. Save existing messages?.. NO_




     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **LABEL**.

```
BLM1TUCU                   FUNCTION LINE SUMMARY                    LINE 1 OF 8

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

 0. ********* ******** ******************************** **** **** ********
 1. LABEL     UPDATEPR
 2. TRACE
 3. ADDDATA            3,2,6,1,SE + PERA/SMITH          NO
 4. PROCESS   ERROR
 5. TESTFLOW  DONE                                     NO   NO
 6. TESTFIELD JUSTONE  1                               NO   NO   TSCATPLC
 7. ADDDATA            LINECMD UU,DOWN LAST,LINECMD UU  NO
i8. PROCESS   ERROR
*** ********* ******** ******************************** **** **** ********




 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> label
```

On the Label Specification panel (BLM8CU9G) , type **1,update** on the command line and press **Enter**.

```
BLM8CU9G                   LABEL  SPECIFICATION                PANEL: UPDATEPR

Enter 'LABEL' control data; cursor placement or input line entry allowed.



       1. Label name... ________
       2. Literal data. _____________________________



     When you finish, type END to save or CANCEL to discard any changes.









===> 1,update
```

Because *update* may be data or may be a command, the Response Type panel (BLG00100) might appear, depending on your installation.

To indicate that *update* is data, type **2** and press **Enter**.

```
     + BLG00100 --------------- RESPONSE TYPE ----------------- COMMAND-+
     |                                                                   |
     |     RESPONSE TO THE PREVIOUS PANEL COULD BE A COMMAND OR DATA     |
     |                                                                   |
     |  OPTIONS:                                                         |
     |                                                                   |
     |        1. COMMAND....Treat the response as a command.             |
     |        2. DATA.......Treat the response as data (not a command).  |
     |        3. RETURN.....Return to the assisted-entry panel,          |
     |                       NOTE: Remaining replies will be ignored.    |
     |                                                                   |
     |                                                                   |
     |                                                                   |
     +------------------------ SELECT OPTION --------------------------+









BLG03085I The assisted-entry panel response UPDATE looks like a command.
===> 2
```

Type **end** on the command line and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9G                  LABEL  SPECIFICATION               PANEL: UPDATEPR

Enter 'LABEL' control data; cursor placement or input line entry allowed.




       1. Label name... UPDATE__
       2. Literal data. _____________________________




     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **ADDDATA**.

```
BLM1TUCU                   FUNCTION LINE SUMMARY                    LINE 1 OF 9

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

 0. ********* ******** ******************************** **** **** ********
 1. LABEL     UPDATEPR
 2. TRACE
 3. ADDDATA           3,2,6,1,SE + PERA/SMITH          NO
 4. PROCESS   ERROR
 5. TESTFLOW  DONE                                     NO   NO
 6. TESTFIELD JUSTONE  1                               NO   NO   TSCATPLC
 7. ADDDATA           LINECMD UU,DOWN LAST,LINECMD UU  NO
 8. PROCESS   ERROR
i9. LABEL     UPDATE
*** ********* ******** ******************************** **** **** ********



 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> adddata
```

On panel BLM8CU9A, type:

```
2,1,jones,,9
```

directly into the **Literal data** field and press **Enter**. Then, type **end** on the command line and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9A                 ADDDATA SPECIFICATION                PANEL: UPDATEPR

Enter 'ADDDATA' control data; cursor placement or input line entry allowed.



 1. Structured word index.... ____            Structured word.. __________
 2. Get variable data........ NO_             Word acronym..... _______________

 3. Literal data.............. 2,1,JONES,,9_____________________



     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **PROCESS**.

```
BLM1TUCU                  FUNCTION LINE SUMMARY                  LINE 1 OF 10

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

 0. ********* ******** ******************************** **** **** ********
 1. LABEL     UPDATEPR
 2. TRACE
 3. ADDDATA            3,2,6,1,SE + PERA/SMITH          NO
 4. PROCESS   ERROR
 5. TESTFLOW  DONE                                     NO   NO
 6. TESTFIELD JUSTONE  1                               NO   NO   TSCATPLC
 7. ADDDATA            LINECMD UU,DOWN LAST,LINECMD UU  NO
 8. PROCESS   ERROR
 9. LABEL     UPDATE
i0. ADDDATA            2,1,JONES,,9                     NO
*** ********* ******** ******************************** **** **** ********


 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> process
```

On panel BLM8CU9H, type **1,error** on the command line and press **Enter**. Then, type **end** and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9H                 PROCESS SPECIFICATION                 PANEL: UPDATEPR

Enter 'PROCESS' control data; cursor placement or input line entry allowed.



                1. Error label name......<R> ERROR___
                2. Save existing messages?.. NO_




     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **TESTFLOW**.

```
BLM1TUCU                   FUNCTION LINE SUMMARY                    LINE 1 OF 11

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

 0. ********* ******** ******************************** **** **** ********
 1. LABEL     UPDATEPR
 2. TRACE
 3. ADDDATA            3,2,6,1,SE + PERA/SMITH          NO
 4. PROCESS   ERROR
 5. TESTFLOW  DONE                                     NO   NO
 6. TESTFIELD JUSTONE  1                               NO   NO   TSCATPLC
 7. ADDDATA            LINECMD UU,DOWN LAST,LINECMD UU  NO
 8. PROCESS   ERROR
 9. LABEL     UPDATE
10. ADDDATA            2,1,JONES,,9                     NO
i1. PROCESS   ERROR
*** ********* ******** ******************************** **** **** ********


 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> testflow
```

On panel BLM8CU9K type:

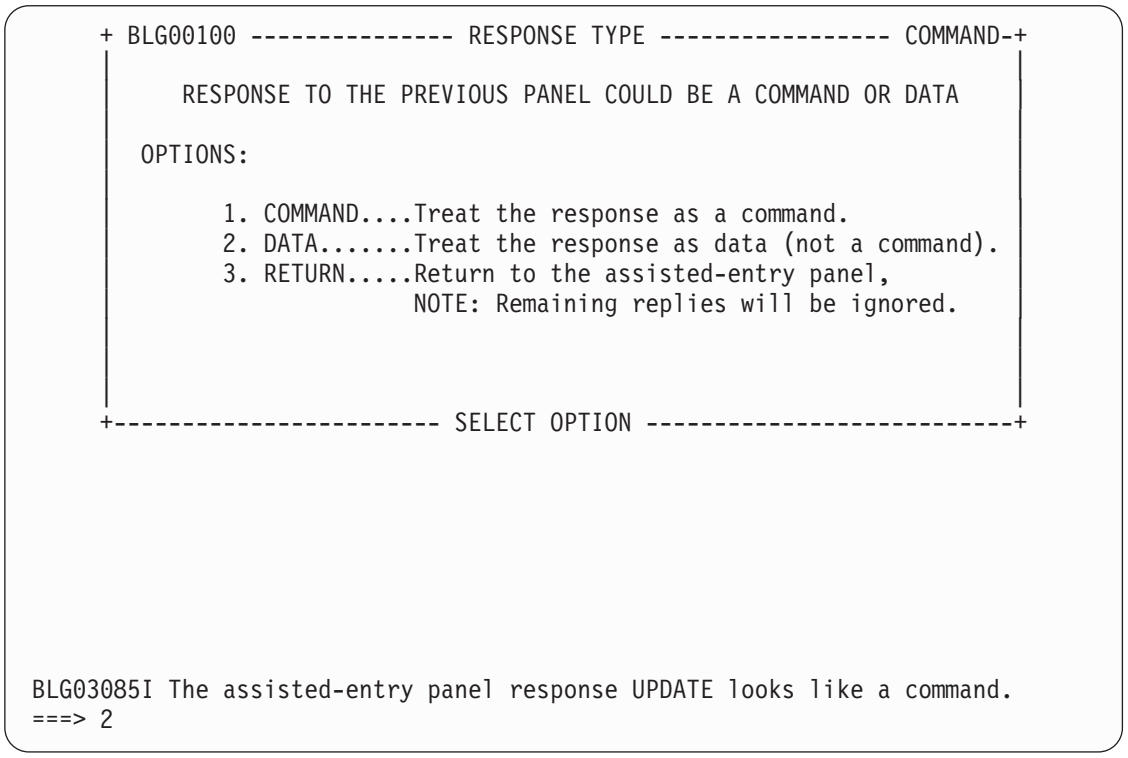
```
1,blg0bu00,3,update
```

on the command line and press **Enter**.

```
BLM8CU9K                 TESTFLOW SPECIFICATION                PANEL: UPDATEPR

Enter 'TESTFLOW' control data; cursor placement or input line entry allowed.



                   1. Verify name.......... ________
                   2. Verify type........<R> PANEL__
                   3. True label.........<R> ________
                   4. Get variable data..... NO_
                   5. Apply not logic....... NO_



     When you finish, type END to save or CANCEL to discard any changes.







===> 1,blg0bu00,3,update
```

Because *update* is also a command, the Response Type panel (BLG00100) might appear, depending on your installation.

To indicate that *update* is data, type **2** and press **Enter**.


+ BLG00100 --------------- RESPONSE TYPE ----------------- COMMAND-+
RESPONSE TO THE PREVIOUS PANEL COULD BE A COMMAND OR DATA
OPTIONS:
1. COMMAND....Treat the response as a command.
2. DATA.......Treat the response as data (not a command).
3. RETURN.....Return to the assisted-entry panel,
NOTE: Remaining replies will be ignored.
+------------------------ SELECT OPTION ---------------------------+
BLG03085I The assisted-entry panel response UPDATE looks like a command.
===> 2

Type **end** on the command line and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9K                TESTFLOW SPECIFICATION               PANEL: UPDATEPR

Enter 'TESTFLOW' control data; cursor placement or input line entry allowed.



                   1. Verify name........... BLG0BU00
                   2. Verify type........<R> PANEL__
                   3. True label.........<R> UPDATE
                   4. Get variable data..... NO_
                   5. Apply not logic....... NO_



     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **BRANCH**.

```
BLM1TUCU                   FUNCTION LINE SUMMARY                   LINE 1 OF 12

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

 0. ********* ******** ******************************** **** **** ********
 1. LABEL     UPDATEPR
 2. TRACE
 3. ADDDATA           3,2,6,1,SE + PERA/SMITH          NO
 4. PROCESS   ERROR
 5. TESTFLOW  DONE                                     NO   NO
 6. TESTFIELD JUSTONE  1                               NO   NO   TSCATPLC
 7. ADDDATA           LINECMD UU,DOWN LAST,LINECMD UU  NO
 8. PROCESS   ERROR
 9. LABEL     UPDATE
10. ADDDATA           2,1,JONES,,9                     NO
11. PROCESS   ERROR
i2. TESTFLOW  UPDATE                                  NO   NO


 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> branch
```

On the BRANCH Specification panel (BLM8CU9B), type **1,done** on the command line and press **Enter** to specify the branch label. Then, type **end** and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9B                  BRANCH  SPECIFICATION                PANEL: UPDATEPR

Enter 'BRANCH' control data; cursor placement or input line entry allowed.


                1. Label name.............<R> DONE____




     When you finish, type END to save or CANCEL to discard any changes.




===> end
```

To insert after the BRANCH control line, you must scroll down. Type **down page** on the command line and press **Enter**.

```
BLM1TUCU                    FUNCTION LINE SUMMARY                    LINE 1 OF 13

   FUNCTION    LABEL   LITERAL                           GET APPLY  FIELD
     NAME      NAME    DATA                              VAR  NOT   NAME

 0. ********* ******** ********************************* **** **** ********
 1. LABEL     UPDATEPR
 2. TRACE
 3. ADDDATA            3,2,6,1,SE + PERA/SMITH           NO
 4. PROCESS   ERROR
 5. TESTFLOW  DONE                                       NO   NO
 6. TESTFIELD JUSTONE  1                                 NO   NO   TSCATPLC
 7. ADDDATA            LINECMD UU,DOWN LAST,LINECMD UU   NO
 8. PROCESS   ERROR
 9. LABEL     UPDATE
10. ADDDATA            2,1,JONES,,9                      NO
11. PROCESS   ERROR
12. TESTFLOW  UPDATE                                     NO   NO


 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> down page
```

Add the next TSP control line, **LABEL**.

```
BLM1TUCU                   FUNCTION LINE SUMMARY                  LINE 13 OF 13

   FUNCTION    LABEL   LITERAL                            GET APPLY  FIELD
     NAME      NAME    DATA                               VAR  NOT   NAME

i3. BRANCH    DONE
*** ********* ******** ******************************** **** **** ********









 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> label
```

On panel BLM8CU9G, type **1,justone** on the command line and press **Enter**. Then, type **end** and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9G                   LABEL  SPECIFICATION               PANEL: UPDATEPR

Enter 'LABEL' control data; cursor placement or input line entry allowed.




       1. Label name... JUSTONE_
       2. Literal data. ____________________________________




     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **ADDDATA**.

```
BLM1TUCU                  FUNCTION LINE SUMMARY                 LINE 13 OF 14

   FUNCTION    LABEL   LITERAL                            GET APPLY  FIELD
     NAME      NAME    DATA                               VAR  NOT   NAME

13. BRANCH    DONE
i4. LABEL     JUSTONE
*** ********* ******** ********************************** **** **** ********









 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> adddata
```

On panel BLM8CU9Am type

```
linecmd u,2,1,jones,,9
```

into the **Literal data** field and press **Enter**. Then, type **end** on the command line and press **Enter** to return to panel BLM8TUCU.

```
BLM8CU9A                ADDDATA SPECIFICATION                PANEL: UPDATEPR

Enter 'ADDDATA' control data; cursor placement or input line entry allowed.



 1. Structured word index.... ____            Structured word.. __________
 2. Get variable data........ NO_             Word acronym..... ________________

 3. Literal data.............. LINECMD U,2,1,JONES,,9__________


     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **PROCESS**.

```
BLM1TUCU                   FUNCTION LINE SUMMARY                  LINE 13 OF 15

   FUNCTION    LABEL   LITERAL                           GET APPLY  FIELD
     NAME      NAME    DATA                              VAR  NOT   NAME

13. BRANCH    DONE
14. LABEL     JUSTONE
i5. ADDDATA             LINECMD U,2,1,JONES,,9           NO
*** ********* ******** ******************************** **** **** ********









 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> process
```

On panel BLM8CU9H, type **1,error** on the command line and press **Enter**. Then, type **end** and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9H                 PROCESS SPECIFICATION                PANEL: UPDATEPR

Enter 'PROCESS' control data; cursor placement or input line entry allowed.



                 1. Error label name......<R> ERROR___
                 2. Save existing messages?.. NO_







     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **LABEL**.

```
BLM1TUCU                  FUNCTION LINE SUMMARY                 LINE 13 OF 16

   FUNCTION    LABEL   LITERAL                            GET APPLY  FIELD
     NAME      NAME    DATA                               VAR  NOT   NAME

13. BRANCH    DONE
14. LABEL     JUSTONE
15. ADDDATA             LINECMD U,2,1,JONES,,9           NO
i6. PROCESS   ERROR
*** ********* ******** ******************************** **** **** ********









 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> label
```

On panel BLM8CU9G, type **1,done** on the command line and press **Enter**.

Then, type **end** and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9G                  LABEL  SPECIFICATION               PANEL: UPDATEPR

Enter 'LABEL' control data; cursor placement or input line entry allowed.




       1. Label name... DONE____
       2. Literal data. _____________________________________




     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **RETURN**.

```
BLM1TUCU                  FUNCTION LINE SUMMARY                 LINE 13 OF 17

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

13. BRANCH    DONE
14. LABEL     JUSTONE
15. ADDDATA           LINECMD U,2,1,JONES,,9           NO
16. PROCESS   ERROR
i7. LABEL     DONE
*** ********* ******** ******************************** **** **** ********









 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> return
```

Because RETURN takes no parameters, no specification panel is displayed.

Add the next TSP control line, **LABEL**.

```
BLM1TUCU                  FUNCTION LINE SUMMARY                 LINE 13 OF 18

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

13. BRANCH    DONE
14. LABEL     JUSTONE
15. ADDDATA           LINECMD U,2,1,JONES,,9           NO
16. PROCESS   ERROR
17. LABEL     DONE
i8. RETURN
*** ********* ******** ******************************** **** **** ********









 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> label
```

On panel BLM8CU9G, type **1,error** on the command line and press **Enter**.

Then, type **end** and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9G                   LABEL  SPECIFICATION                PANEL: UPDATEPR

Enter 'LABEL' control data; cursor placement or input line entry allowed.




       1. Label name... ERROR___
       2. Literal data. _____________________________




     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **PRINT**.

```
BLM1TUCU                   FUNCTION LINE SUMMARY                  LINE 13 OF 19

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

13. BRANCH    DONE
14. LABEL     JUSTONE
15. ADDDATA             LINECMD U,2,1,JONES,,9           NO
16. PROCESS   ERROR
17. LABEL     DONE
18. RETURN
i9. LABEL     ERROR
*** ********* ******** ******************************** **** **** ********




 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> print
```

On the Print Specification panel (BLM8CU9R), type

```
1,yes,2,yes,3,yes
```

to specify that you want all information printed.

Type **end** and press **Enter** to return to panel BLM1TUCU.

```
BLM8CU9R                PRINT  SPECIFICATION                PANEL: UPDATEPR

Enter 'PRINT' control data; cursor placement or input line entry allowed.



                  1. Print the messages........ YES
                  2. Print the screen.......... YES
                  3. Print the TSCA............ YES




     When you finish, type END to save or CANCEL to discard any changes.









===> end
```

Add the next TSP control line, **RETURN**.

```
BLM1TUCU                    FUNCTION LINE SUMMARY                  LINE 13 OF 20

   FUNCTION    LABEL   LITERAL                            GET APPLY  FIELD
     NAME      NAME    DATA                               VAR  NOT   NAME

13. BRANCH    DONE
14. LABEL     JUSTONE
15. ADDDATA             LINECMD U,2,1,JONES,,9           NO
16. PROCESS   ERROR
17. LABEL     DONE
18. RETURN
19. LABEL     ERROR
i0. PRINT
*** ********* ******** ******************************** **** **** ********




 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> return
```

To complete the TSP, type **end** on the command line and press **Enter**.

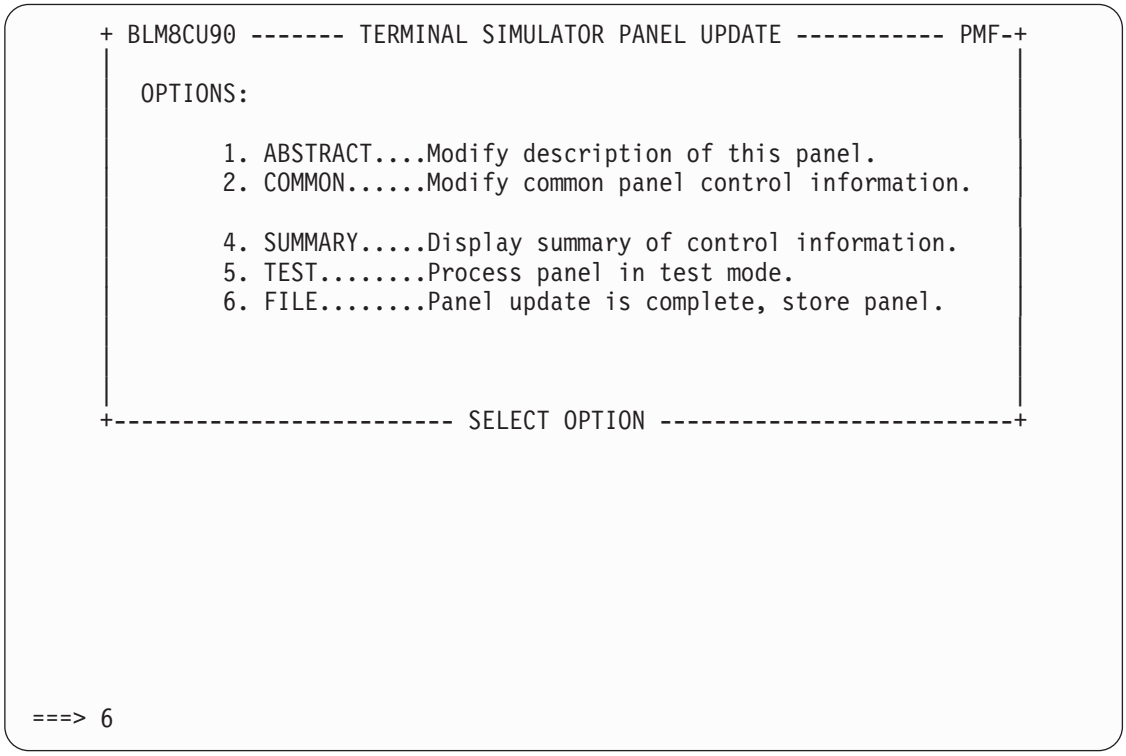
```
BLM1TUCU                   FUNCTION LINE SUMMARY                  LINE 13 OF 21

   FUNCTION    LABEL   LITERAL                          GET APPLY  FIELD
     NAME      NAME    DATA                             VAR  NOT   NAME

13. BRANCH    DONE
14. LABEL     JUSTONE
15. ADDDATA             LINECMD U,2,1,JONES,,9          NO
16. PROCESS   ERROR
17. LABEL     DONE
18. RETURN
19. LABEL     ERROR
20. PRINT
21. RETURN
*** ********* ******** ******************************** **** **** ********




 Line Cmds:  A=After  C=Copy  D=Delete  I=Insert  M=Move  R=Repeat  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit.

===> end
```

Type **end** and press **Enter** again.

```
BLM8CU91               TERMINAL SIMULATOR PANEL UPDATE             EXTERNALS

+-------------------------------------------------------------------------------+
| UPDATEPR                                                                  PMF |
|                                                                               |
|  This TSP will search the problem records for problems assigned to Smith.     |
|  It will then reassign these problems to Jones.                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
+-------------------------------------------------------------------------------+

   Modify textual data within the box.  To modify control data, type
   CONTROL on the command line.  When you finish, type END to save or
   CANCEL to discard any changes.




===> end
```

To file the TSP, type **6** on the command line and press **Enter**.

This option writes the TSP to your write panel data set. For more information about this panel, see "Using the Terminal Simulator Panel Update Panel (BLM8CU90)" on page 48.



After you file your TSP, you return to the Panel Name Entry panel where you started. The message tells you that your TSP has been filed in your write panel data set.

To leave the panel, type **end** and press **Enter**.

```
BLM8CU00                     PANEL NAME ENTRY                              UPDATE

Identify panel to be updated; cursor placement or input line entry allowed.




               1. Panel name.................<R> UPDATEPR
               2. Data set definition label..... ________




 To enter the panel update dialog, press Enter without field modifications.







BLM04015I Panel UPDATEPR was written to the WRITE panel data set. Processing c+
===>
```

Copy the TSP to a read panel data set before running it. For information on copying panels, refer to the *Tivoli Information Management for z/OS Panel Modification Facility Guide*. For information on how to run a TSP, see "Running Terminal Simulator Panels (TSPs) and Terminal Simulator Execs (TSXs)" on page 257.

Using the Terminal Simulator Panel Update Panel (BLM8CU90)

After you exit panel BLM8CU91, panel BLM8CU90 is displayed. This panel gives you the following options:

Abstract
: To change (insert, delete, update, copy, or move) control lines in an existing TSP and modify the descriptive text of the TSP.

Common
: To limit the starting of a TSP to a single panel.

Summary
: To change (delete or update) control lines in a TSP.

Test To test a TSP in an interactive environment.

File To save your TSP in your write panel data set.

Except for Option 2, **Common**, these options are similar in function to the options on other Tivoli Information Management for z/OS update panels and are not discussed here. The *Tivoli Information Management for z/OS Panel Modification Facility Guide* contains additional information about these options.

You can use **Common** to ensure that your TSP can only be started when a specific panel is the current panel.

Note: Be careful when using **Common**. Specify this option only if your TSP must only be started from a specific panel. Do not specify a starting panel if you want to run your TSP from more than one panel. As an alternative, you can use the TESTFLOW control line. For more information about the TESTFLOW control line, see "TESTFLOW" on page 180.

To specify a starting panel from panel BLM8CU90, type **2** and press **Enter**. On the Common Update panel BLM8CU97, enter a panel name in the **Starting panel name** field.

```
BLM8CU97                        COMMON UPDATE                      PANEL: ________

Enter common panel control data; cursor placement or input line entry allowed.



 1. User service level....... ________       Communicating panel.. ___
 2. Starting panel name...... ________       Dialog section....... __
                                             IBM release level.... __
                                             IBM PTF level........ __
                                             IBM FMID/APAR level.. ________
                                             Panel modified....... ___
                                             Date last altered.... __________
                                             Time last altered.... _____
                                             User last altered.... ________

     When you finish, type END to save or CANCEL to discard any changes.




===>
```

If the name entered in the **Starting panel name** field does not match that of the current Tivoli Information Management for z/OS panel at the time you try to run the TSP, Tivoli Information Management for z/OS issues an error message and does not start the TSP. For more information on how a starting panel name can affect the running of a TSP, see Running Terminal Simulator Panels (TSPs) and Terminal Simulator Execs (TSXs).

The *Tivoli Information Management for z/OS Integration Facility Guide* is a good source of further examples of TSPs and TSP control line usage; it contains a list of all Integration Facility TSPs and what they do.

Examples: Adding or Updating Freeform Text

You can use the ADDTEXT and REPTEXT TSX control lines to add or update text.

You can also add or update freeform text by using a combination of USEREXIT, ADDDATA, and PROCESS control lines in your TSP:

- Use the USEREXIT control line to change the current table panel line field in the TSCA.
- Use the ADDDATA control line to insert scroll commands, the FIND command, or the LINECMD command into the command line reply buffer.
- Use the PROCESS control line to run the command that is in the reply buffer. A new line becomes the current table panel line.

Note: If you want to add or update freeform text in a TSP, you must use the Tivoli Information Management for z/OS editor. You must set this condition in the user profile. For information about user profiles, refer to the *Tivoli Information Management for z/OS User's Guide*.

For example, assume that previous TSP processing brought a problem record in for update and the current panel is the Problem Text Entry panel (BLG0B010). Here, you select the type of freeform text that you want to enter. To ensure that the TSP uses the Tivoli Information Management for z/OS editor to enter freeform text, the TSP sets your profile so that the **Use Info editor for SRCs and TSPs?** field is **YES**, then it selects the type of freeform text that you want to enter. At this point, the current panel is a table panel, and the current line is the first line of text. The TSP then uses the ADDDATA, PROCESS, and USEREXIT control lines to add text to the bottom of the freeform text. When it finishes, the TSP files the problem record and resets your profile to its original values.

This example shows one way this can be written. The record must already be in update mode. From a problem summary panel, this TSP adds a line of freeform text when it is used with the correct user exit. The TSP can be started using the RUN command and the TSP name (in this example, USRTSP32). You can also set up an alias name for the TSP by adding an entry to the ALIAS record. This record correlates the alias name to the TSP. The *Tivoli Information Management for z/OS Program Administration Guide and Reference* contains additional information about the ALIAS record.

```
LABEL       USRTSP32
ADDDATA               SUS,
ADDDATA               PROFILE,1,52,YES,END,9
ADDDATA               RES,
PROCESS     ERROR
   ( ABOVE SECTION SETS PROFILE TO USE THE INFO EDITOR )

ADDDATA               8,1
PROCESS     ERROR
   ( ENTER INTO FREEFORM DESCRIPTION TEXT PANEL )

ADDDATA               DO LAST,UP 1,
PROCESS     ERROR
   ( SET CURRENT LINE FOR THE TESTFIELD THAT FOLLOWS )

LABEL       LOOPTOP
TESTFIELD   LOOPEND   0                 TSCACTBL         YES
   ( CHECK FOR TSCACTBL=0, IF THERE IS TEXT THEN GO TO LOOPEND)
   ( IF THERE IS NO TEXT THEN LOOP TILL IT FINDS TEXT )
   ( NOT LOGIC IS SET TO YES )

ADDDATA               UP 1,
PROCESS     ERROR
BRANCH      LOOPTOP  ( UNCONDITIONAL BRANCH )
   ( END OF LOOP )
```

```
LABEL       LOOPEND
ADDDATA               LINECMD I,
ADDDATA               DO 1,
PROCESS     ERROR
USEREXIT                                TBLDATA          NEXT     NO
   ( THIS SECTION ADDS THE FREEFORM TEXT )

ADDDATA               END,END,PROF,8
PROCESS     ERROR
   ( RESETS THE PROFILE TO THE LAST PERMANENTLY SAVED STATUS )

LABEL       ERROR
RETURN
```

This example also adds freeform text to a record. In “Assembler Code User Exit Example” on page 303, you can find a user exit routine written in assembler language that works with this TSP. The record is already in update mode or create mode.

```
 1  LABEL     ADDFFTXT
 2  ADDDATA             PROFILE,1,52,YES,END,9                        NO
 3  PROCESS   ERROR
                      ( SET USER PROFILE TO USE INFO EDITOR )
 4  ADDDATA             8,1                                           NO
 5  PROCESS   ERROR
                      ( ENTER FREEFORM DESCRIPTION TEXT )
 6  ADDDATA             LINECMD I,DO 1                                NO
 7  PROCESS   ERROR
                      ( MOVE CURSOR TO CORRECT LINE )
 8  USEREXIT                                    TBLDATA   NEXT    NO    NO
                      ( CALL USEREXIT WITH DATA TO BE ADDED )
 9  ADDDATA             END,END,PROF,8                                NO
10  PROCESS   ERROR
                      ( FILE RECORD WITH FREEFORM TEXT ADDED AND
                                RESET THE USER PROFILE )
11  LABEL     ERROR
12  RETURN
```

3

Designing and Creating a Terminal Simulator EXEC (TSX)

Overview

A TSX is a REXX EXEC that can be run instead of a TSP anywhere that a TSP can run. Refer to "Running Terminal Simulator Panels (TSPs) and Terminal Simulator Execs (TSXs)" on page 257 for additional information on all of the methods that you can use to invoke TSPs and TSXs.

A TSX, like a TSP, enables you to perform Tivoli Information Management for z/OS tasks. The intent of the TSX is to allow a REXX user to customize Tivoli Information Management for z/OS without having to learn PMF and TSP coding skills.

A callable routine, BLGTSX, is provided that the TSX can call to perform the function of most TSP control lines. Control lines define functions to Tivoli Information Management for z/OS. As with a TSP, you use control lines in a TSX to define what you want to do. In most instances, control lines that are available to a TSP are available to a TSX. For example, the control line FINDSDATA is of importance to both the TSP user and the TSX user and is therefore made available to TSPs and TSXs. However, the control line TESTFIELD is available only to a TSP user because a TSCA field value can be tested directly with REXX in a TSX. Refer to "Creating Terminal Simulator Control Lines" on page 61 for a complete list of the control lines that are available in a TSP, a TSX, or both.

TSX control lines, keyword values, and some parameter values can be entered in lower or mixed case. To see whether a parameter can be entered in lower or mixed case, see the specific control line in "Creating Terminal Simulator Control Lines" on page 61.

In addition, you can pass an argument to a TSX (via the RUN command or via the LINK control line) which the TSX can parse into parameters as needed. This feature enables you to write a single TSX that performs actions that may vary, based on the parameters, rather than writing many similar TSXs. For example, a TSX to flatten a record can be passed the RNID of the record to be flattened. The *Tivoli Information Management for z/OS User's Guide* contains additional information on the RUN command. For additional information on the LINK control line, refer to "LINK" on page 130.

If you have ever written a TSP, you know that even simple tasks like using a counter to control a loop can be difficult. With a TSX, you have all of the functions of a high level language available to you.

TSX Control Lines

Within a TSX, Tivoli Information Management for z/OS services (that is, control lines) are performed by calling module BLGTSX. Parameters are provided with the call to indicate the selected control line, options, and data. Tivoli Information Management for z/OS also sets REXX variables with the values of the following TSCA fields when the TSX is initialized and after each call to BLGTSX (note that the variable names are the same as the names of the TSCA fields):

TSCACMOF Decimal	**TSCARSD Character**
TSCACMRB Character	**TSCASDF Character**
TSCACPNL Character	**TSCASESS Character**
TSCACRID Character	**TSCASUBP Pointer**
TSCACTBL Decimal	**TSCASUSP Decimal**
TSCAFRES Decimal	**TSCATBLL Character**
TSCAFRET Decimal	**TSCATLIX Decimal**
TSCALFDB Character	**TSCATPLC Decimal**
TSCALFID Character	**TSCATPLN Decimal**
TSCAMTBL Decimal	**TSCAUFLD Character**
TSCAPRIV Character	**TSCAUPTR Pointer**
TSCARPD Character	**TSCAVDA (contents of variable area) Character**
TSCARRB Character	**TSCAVPH Character**

Character format fields will have trailing blanks and hexadecimal zeros removed. *Binary* fields will be converted to decimal in character format with no leading zeros. *Pointer* fields will be converted to hexadecimal in character format.

Note: You can use the SETTSCA control line to change the values of the TSCAUFLD, TSCAUPTR, and TSCATLIX fields. Additional information on the SETTSCA control line can be found in “SETTSCA” on page 174. You cannot change the value of any TSCA fields by assigning values to the corresponding REXX variables.

Upon completion of a PROCESS control line, any messages currently on the message chain will be stored in the REXX compound variable **BLG_MESSAGE.** with the count of messages (TSCAMSGC) contained in variable **BLG_MESSAGE.0**. Refer to “Message Checking in a TSX” on page 253 for additional information on checking messages in a TSX.

If a call to BLGTSX has a parameter error (for example, a required parameter not specified), it will cause a REXX syntax error. Messages describing syntax errors are contained in compound variable **BLG_ERROR.**, with the count of errors contained in variable **BLG_ERROR.0**. Refer to "Syntax Checking in a TSX" on page 253 for additional information on syntax checking in a TSX.

To perform a control line function within the TSX, call routine BLGTSX passing the control line name, options, and data (see "Creating Terminal Simulator Control Lines" on page 61 for details). After the processing of each control line, your routine should check the REXX variables associated with the TSCA fields set by the control line to determine the results of the control line processing.

To verify that the TSX is starting on the correct panel (simulating the function of the *Starting panel name* field in a TSP), begin your EXEC with a test of variable TSCACPNL, that contains the current panel name. If TSCACPNL does not contain the correct panel name, code your EXEC to issue a message and exit.

Graphic Character Substitutions using REXX Variable BLGSYMB

The Graphic Character Substitutions feature of Tivoli Information Management for z/OS enables you to use substitute characters for four of the graphic characters that Tivoli Information Management for z/OS uses. The code points for these four characters are X'5F' (the "not" symbol ¬), X'4F' (the "or bar" symbol |), X'5A' (the "exclamation" symbol **!**), and X'7C' (the "at" symbol @), all on code page 37. If you do not use Graphic Character Substitutions, the characters that you use are the ones that are at those code points on your code page. Graphic Character Substitutions lets you specify other characters as substitutes, in which case you would use the substitution characters.

TSXs that code "not" symbols or "or bar" symbols in data that they pass to Tivoli Information Management for z/OS for processing must use the appropriate characters. Rather than coding these characters in your TSXs, your TSXs can retrieve the characters to be used as the "not" symbol and "or bar" symbol from the REXX variable BLGSYMB. This ensures that your TSXs specify the proper characters for the "not" symbol and "or bar" symbol as defined by your installation.

BLGSYMB is a REXX variable that is set when a TSX is initialized. It is a string that contains:

1. The character to be used for the "not" symbol followed by a blank, followed by
2. the character to be used for the "or bar" symbol followed by a blank, followed by
3. the character to be used for the "exclamation" symbol followed by a blank, followed by
4. the character to be used as the "at" symbol

In general, TSXs can ignore everything beyond the "or bar" symbol because the "exclamation" symbol and the "at" symbol do not initiate Tivoli Information Management for z/OS processing when included in data passed from the TSX to Tivoli Information Management for z/OS.

The following example shows how you can code your TSX to parse the "not" symbol and the "or bar" symbol from the REXX variable BLGSYMB. In the example, the "not" symbol and the "or bar" symbol are parsed and saved in variables called *notsign* and *orbar*. Because the TSX does not need the "exclamation" or "at", the rest of the data in BLGSYMB is ignored.

```
parse var BLGSYMB notsign orbar . /* Get the user's not symbol and
                                     or bar symbol from REXX variable BLGSYMB */
```

After parsing BLGSYMB, your TSX should code the variable that contains the parsed "not" symbol in data being passed to Tivoli Information Management for z/OS for processing rather than coding the "not" symbol in that data. Likewise, your TSX should do the same for the "or bar" symbol. This example builds a response that subsequently will be passed to Tivoli Information Management for z/OS for processing. It uses the "not" symbol to remove the data from the fourth column on a list processor table panel. Previously you may have coded the "not" symbol in the response:

```
irc='LINECMD L4,¬'
```

Instead of coding the "not" symbol, which might not be the character to use for "not" symbol processing (depending on your installation), you should code the variable containing the "not" symbol that was parsed from BLGSYMB:

```
irc='LINECMD L4,'notsign
```

You could also do the same for the "or bar" variable. For example:

```
irc='LINECMD L4,'orbar
```

API applications can call the HLAPI extension BLGTSPCH to retrieve the characters to be used as the "not" symbol and the "or bar" symbol in data being passed to Tivoli Information Management for z/OS for processing. For a description of this TSX, refer to the *Tivoli Information Management for z/OS Application Program Interface Guide*.

TSX Access

In order to invoke a TSX, you must allocate a DD statement BLGTSX that points to the data set that contains your REXX EXECs before logging onto Tivoli Information Management for z/OS. Data set **BLM.SBLMTSX** is shipped with Tivoli Information Management for z/OS, and this data set contains the TSXs used by immediate notification. Therefore, the BLGTSX DD should allocate data set **BLM.SBLMTSX**.

Note: You may need to substitute the data set name used at your installation for the BLM.SBLMTSX data set name given above.

You can also concatenate multiple data sets to the BLGTSX DD. Create another data set for user modified or created TSXs, and add it to the BLGTSX DD concatenation. Refer to the *Tivoli Information Management for z/OS Planning and Installation Guide and Reference* for more information on starting Tivoli Information Management for z/OS and allocating the BLGTSX data set where your TSX REXX EXECs reside.

Creating a TSX

This section will walk you through the steps required to create and run a sample TSX. The sample describes how to call BLGTSX with the necessary parameters for the MESSAGE, READDICT, PROCESS, and GETLIST control lines. The sample TSX will show you how the output from the TSX READDICT control line is used as input to the TSX GETLIST control line. You will also see how EXECIO can be used to copy information from your TSX REXX compound variables to a data set. The TSX even takes care of allocating and freeing the output data set so that the contents of the data set can be displayed to the user when the TSX ends. This section also provides some TSX error handling techniques.

Note: Because most data entry and display operations on date fields use external date format, and because different users can use different external date formats, it is

recommended that you use internal date format YYYY/MM/DD for all date processing. Use BLGIDATE to convert any date values retrieved from a record to internal format, then perform any required processing, and then use BLGEDATE to convert an internal date to external date format before entering the date in a field with the PROCESS control line. The user exits BLGIDATE and BLGEDATE are described in "General-purpose User Exits" on page 279.

TSX Objective: A user would like to be able to write the list processor data associated with s-word index S1416 from any given problem record to a data set. The user wants to browse the data set from Tivoli Information Management for z/OS once the list processor data has been written.

You can create a TSX that is passed the RNID of the record that contains the list processor data. The TSX will use the GETLIST control line to retrieve the list processor data. Once GETLIST has retrieved the list processor data, you can use EXECIO to write the list processor data to a data set. The REPORT command has a PRINT/BROWSE facility that will allow the TSX to display the data set which contains the list processor data to the user once the TSX has finished running.

1. Allocate the BLGTSX DD:
   ```
   TSO ALLOC FI(BLGTSX) DA('my.tsx.execs''BLM.SBLMTSX') SHR REUSE
   ```
2. Create a member called **PRTLPTSX** in your TSX EXEC data set.
3. Type in the sample contained in 58.
4. Create or update the ALIAS record so that it contains an entry for the sample TSX, and then file the ALIAS record. In the example below, the alias name is LP.

   Note: This step is optional. You do not have to create an entry for the TSX in the ALIAS record.

```
===>

BLGLALIS                    ALIAS NAME ENTRY                      LINE 1 OF 22

USE....List alias name, actual name, and type (panel or REXX EXEC).
                                                                  RECORD: ALIAS


         ALIAS      ACTUAL                          ALIAS      ACTUAL
         NAME       NAME      TYPE                  NAME       NAME      TYPE
  ''   LP______   PRTLPTSX   X          '''''   ________   ________   _
  ''   ________   ________   _          '''''   ________   ________   _
  ''   ________   ________   _          '''''   ________   ________   _
  ''   ________   ________   _          '''''   ________   ________   _
  ''   ________   ________   _          '''''   ________   ________   _
  ''   ________   ________   _          '''''   ________   ________   _
  ''   ________   ________   _          '''''   ________   ________   _
  ''   ________   ________   _          '''''   ________   ________   _
  ''   ________   ________   _          '''''   ________   ________   _
  ''   ________   ________   _          '''''   ________   ________   _
  ''   ________   ________   _          '''''   ________   ________   _
 Line Cmds:  A=After  B=Before  C=Copy  D=Delete  E=Erase  I=Insert
             L=Line entry  M=Move  R=Repeat
 Type DOWN, UP, LEFT, or RIGHT to scroll the panel, or type END to exit.
```

5. Verify that the REXX variable PNLNAME in the sample in 58 is assigned the name of your problem record summary panel. The default is BLG0BU00.
6. Allocate the data set that you specify for the variable MYDSNAME in the sample in 58. The default is **MARYANN.TSX.OUTPUT**.
7. Create a record on your database that contains list processor data that is associated with s-word index S1416. If you choose, you may use a different s-word index in the sample. Also, remember the RNID of the record. For this example, assume that the RNID is 46.

   Note: S-word index S1416 is associated with problem record symptom data, specifically with device names. The list processor panel is BLGLDEVL, and the assisted-entry panel is BLG6DEVL. From the problem record summary panel, BLG0BU00, choose selection **4. Symptom data** to flow to the problem symptom data panel, BLG0B402, and then choose selection **2. Device names** to flow to the list processor panel, BLGLDEVL.
8. In step 4, it was suggested that you create an ALIAS record for the member PRTLPTSX. If you did so, run the TSX passing the RNID of the record that you created in step 7. Type **RUN LP 46** on the Tivoli Information Management for z/OS command line, passing parameter 46, the RNID of the record that you created in step 7. (If you chose not to create an ALIAS record, type **RUN PRTLPTSX 46** on the command line to pass parameter 46.)
9. The list processor data will be displayed to the user in a data set in BROWSE mode. When the user exits the data set, the panel from which the TSX was run is displayed.

This TSX retrieves list processor data associated with a record and writes it to a data set.

```
/*********************************************************************/
/* This TSX is passed an RNID on invocation.  The TSX then uses      */
/* the GETLIST control line to retrieve the List Processor data      */
/* associated with the record.  If the retrieve is successful, EXECIO*/
/* is used to print the List Processor data to a data set.           */
/*********************************************************************/
SIGNAL ON SYNTAX                  /* Call SYNRTN if any syntax or TSX*/

exec_fmid='HOYB100'               /* Set current maintenance level   */
parse source . . execname .       /* Get exec name                   */
if blgtrace=1 then                /* TSP/TSX tracing requested?      */
  do                              /* Enable tracing                  */
    say '**** Entering' left(execname,8) '(level='exec_fmid') ****'
    trace_option='Results'        /* Set trace option                */
  end                             /* Enable tracing                  */
else                              /* TSP/TSX tracing not requested   */
  trace_option='Normal'           /* Use default trace option        */
trace (trace_option)              /* Use requested trace option      */

parse var BLGSYMB notsign orbar . /* Get local not sign/or bar       */

PARSE SOURCE . . EXECNAME .     /* Get EXEC name.                    */

MYDSNAME="'MARYANN.TSX.OUTPUT'";  /* The output data set name.       */

PARSE ARG RNID .           /* Get the RNID passed on the RUN command */

IF RNID='' THEN,           /* Is there an RNID?                      */
   DO;
      MSGTEXT='This TSX requires an RNID on the RUN command.';
      CALL BLGTSX 'MESSAGE',,MSGTEXT;
      EXIT;
   END;
```

```
RESPONSE=';UPDATE R' RNID;  /* RESPONSE contains the command to      */
                            /* UPDATE the RNID.                      */

CALL BLGTSX 'PROCESS',RESPONSE,'DISCARD'; /* UPDATE the record and   */
                                          /* DISCARD messages.       */

IF TSCAFRET/=0 THEN,       /* Was the problem record successfully    */
                           /* updated?                               */
   DO;
      EXIT;
   END;

PNLNAME='BLG0BU00';
IF TSCACPNL=PNLNAME THEN,    /* The RNID was successfully updated,   */
                             /* and the record is a problem record.  */
   DO;
      SWORD='S1416';
      CALL BLGTSX 'READDICT',SWORD;   /* Get s-word from dictionary  */
      /* The s-word will be stored in TSCARSD.                       */
      ROOTSWORD=TSCARSD;           /* The root s-word of the LP data.*/
      BLG_LIST.  = '';             /* Initialize the array.          */
      BLG_LIST.0 =  0;        /* Initialize the # of array elements. */
      CALL BLGTSX 'GETLIST',ROOTSWORD;
      /* The LP data is stored in REXX compound variable BLG_LIST.   */
      /* The number of elements in the array is stored in BLG_LIST.0.*/
   END;
ELSE
   DO;
      RESPONSE=';CANCEL';               /* CANCEL out of the record. */
      CALL BLGTSX 'PROCESS',RESPONSE,'DISCARD';
      MSGTEXT='The summary panel is not' PNLNAME'.';
      CALL BLGTSX 'MESSAGE',,MSGTEXT;
      EXIT;
   END;

IF TSCAFRET=0 & TSCAFRES=0 THEN, /* GETLIST successful */
   DO;
   /* Write the list data to a data set.  The data set is defined as */
   /* RECFM=VB                                                       */
      "ALLOC FI(MYDD) DA("MYDSNAME")";
      'EXECIO * DISKW' MYDD '(FINIS STEM BLG_LIST.)'
      "FREE FI(MYDD)";
      DSNAME=STRIP(MYDSNAME,'B',"'");     /* Get rid of quotes.      */
      /* Browse the LP data.                                         */
      RESPONSE=';CANCEL,;REPORT,10,1,'DSNAME',2,BROWSE,3,NO,,';
      CALL BLGTSX 'PROCESS',RESPONSE,'DISCARD';
   END;
ELSE
   DO;
      RESPONSE=';CANCEL';                 /* Put the user back on the
                                        panel that they started on.  */
      CALL BLGTSX 'PROCESS',RESPONSE,'DISCARD';
      MSGTEXT='The record does not contain LP data for S1416.';
      CALL BLGTSX 'MESSAGE',,MSGTEXT;
   END;


if blgtrace=1 then               /* TSP/TSX tracing requiested?     */
  say '**** Leaving ' left(execname,8 '(level='exec_fmid') ****'

EXIT

/* Subroutine to issue a message                                     */
/* arg(1)=Message Type, arg(2)=Message ID, arg(3..n)=message inserts */
issuemsg:
  if arg(1)='SAY' then            /* If message type is SAY          */
```

```
    msgtype='BUILD'              /*   treat as BUILD for ctl line   */
  else                           /* A message type other than SAY   */
    msgtype=arg(1)               /* Arg 1 is message type           */
  insert.1=execname              /* Use TSX name as parameter 1     */
  inscnt=1                       /* TSX name is insert number 1     */
  do argno=3 to arg()            /* Copy args to insert. stem       */
  do argno=3 to arg()            /* Copy args to insert. stem       */
    inscnt=inscnt+1              /* Increment insert count          */
    insert.inscnt=arg(argno)     /* Copy next arg to insert. var    */
  end                            /* Set insert. stem values         */
  CALL BLGTSX 'MESSAGE',arg(2),,msgtype,,'INSERT.',inscnt
  if arg(1)='SAY' then           /* If message type is SAY          */
    say tscavda                  /* Output the message              */
  return

/* Subroutine to display helpful information in the event of a syntax */
/* or TSX control line parameter error                                */
syntax:
  errsigl=sigl                        /* Save failing line number    */
  call issuemsg 'SAY',20200,sigl      /* Show failing line number    */
  say strip(SourceLine(errsigl),'T')  /*   and the line source       */
  if symbol('BLG_ERROR.0')='VAR' then /* Control line errors?        */
    do i = 1 to BLG_ERROR.0           /* Loop through the messages   */
      say BLG_ERROR.i                 /* Display error message       */
    end
  /* If API active then set syntax error reason code                 */
  if blgapi=1 then
    do
      Call BLGTSX 'SetAPIData','HICARETC',12
      Call BLGTSX 'SetAPIData','HICAREAS',165
    end
  exit 8
```

4

Creating Terminal Simulator Control Lines

This chapter describes the TSP and TSX control lines. This table shows which control lines are supported by TSPs, which control lines are supported by TSXs, and which control lines are supported by both TSPs and TSXs. When a TSP control line is not available in a TSX, an explanation is provided.

Table 1. Summary of TSP and TSX control lines

Control Line	TSP	TSX	Additional Explanation of TSX Availability
ADDDATA	X		Note 2
ADDLIST		X	
ADDSDATA		X	
ADDTEXT		X	
BRANCH	X		Note 1
CLEAR	X		Note 1
CLOSERRES		X	Note 5
CLOSESOCKET		X	
DELLIST		X	
DELSDATA		X	
DELTEXT		X	
DEQMAIL		X	
FINDSDATA	X	X	
FINDSJRNL	X	X	
FINDTEXT		X	
FLATTEN	X	X	
GETAPIDATA		X	
GETLIST		X	
GETRDATA		X	Note 5
GETSCREEN		X	
GETTEXT		X	
ISPEXEC	X		Note 1
LABEL	X		Note 1
LINK	X	X	
MESSAGE	X	X	
MOVEVAR	X		Note 1
OPENRRES		X	Note 5

Table 1. Summary of TSP and TSX control lines (continued)

Control Line	TSP	TSX	Additional Explanation of TSX Availability
OPENSOCKET		X	
PRINT	X	X	
PROCESS	X	X	
PUTRDATA		X	Note 5
QUERYRRES		X	Note 5
QMAIL		X	
READDICT		X	
READSOCKET		X	
RELEASERRES		X	Note 5
REPLIST		X	
REPTEXT		X	
RETURN	X		Note 1
SETAPIDATA		X	
SETFIELD	X		Note 1
SETRRES		X	Note 5
SETTSCA		X	
TESTFIELD	X		Note 1
TESTFLOW	X		Note 1
TRACE	X		Note 1
UNFLATTEN	X	X	
USEREXIT	X	X	Note 4
WORDFIX	X		Note 3
WRITESOCKET		X	

Note: Explanation Details:

1 Function available with standard program flow controls in the REXX programming language.

2 Incorporated into PROCESS control line.

3 The TSX ADDSDATA control line can be used to perform the WORDFIX-like add function and the DELSDATA can be used to perform the WORDFIX-like delete function. The change function of WORDFIX is not supported in a TSX, so in order to perform the change function, you will need to use the LINK control line to link to a TSP to perform WORDFIX.

4 Supports user exits that do not require that information be passed on the TSP USEREXIT control line panels (BLM8CU9P, BLM8CU9Q).

5 Used in Remote Data Resource Service. These TSXs are described in “Remote Data Resource Terminal Simulator Control Lines” on page 241.

ADDDATA

This control line simulates data responses that you enter on the command line of a panel in interactive mode. When creating the ADDDATA control line, you can specify the responses you want to enter. When the TSP is run, the responses that you specified earlier are chained in the TSCA command line reply buffer (TSCACMRB). When you have data from one or more ADDDATA control lines chained in the command line reply buffer, you need a PROCESS control line following the ADDDATA control lines to act as the Enter key.

In TSX usage, the PROCESS control line provides the function of the combination of the TSP control lines ADDDATA and PROCESS. Refer to “The PROCESS TSX Control Line” on page 155 for additional information on how to perform TSP ADDDATA functions in a TSX.

The ADDDATA control line has many uses. The phone number example described in “Examples” on page 1 is one way you could use it. (Your employee in charge of analyzing problem records has moved and received a new phone number.) Assume that the person who moved is uniquely identified in the problem database by the assignee name of Smith. With this information you can develop an immediate response chain (IRC) that locates all the records that require a phone number change. This is an example of how to use an ADDDATA control line to put this IRC into the command line reply buffer:

```
LABEL     UPDATEPR
ADDDATA   6,1,SE + PERA/SMITH STAC/OPEN   (Entered in literal data field)
PROCESS   ERROR
⋮

RETURN
```

Several variations for creating the ADDDATA control line for this IRC are given later in this section.

Creating an ADDDATA Control Line

Use the following ADDDATA SPECIFICATION panel to create an ADDDATA control line.

```
BLM8CU9A                ADDDATA SPECIFICATION                PANEL: ________

Enter 'ADDDATA' control data; cursor placement or input line entry allowed.



 1. Structured word index.... ____            Structured word.. _______________
 2. Get variable data........ NO_             Word acronym..... _______________

 3. Literal data.............. 6,1,SE + PERA/SMITH STAC/OPEN___



     When you finish, type END to save or CANCEL to discard any changes.









===>
```

General Rules

You must enter information in either the **Structured word index** field or the **Literal data** field, or enter **YES** in the **Get variable data** field.

Field Descriptions

1. Structured word index

Valid reply

The index key of an s-word in the dictionary data set or no reply.

An entry in this field indicates that you want the s-word associated with the index added to the data in the command line reply buffer.

Default

No reply.

2. Get variable data

Valid reply

YES, NO, or no reply.

When you enter **NO** or make no reply, this field has no effect. When you enter **YES**, it indicates that you want the variable data added to the data in the command line reply buffer. If you put **YES** in the **Get variable data** field when entering data in either **Structured word index** or **Literal data**, the s-word or literal data is added to the command line reply buffer first, followed by the variable data.

Default

NO

Restrictions

This field has no restrictions on specifying a value when you create the panel. However, if this field is set to **YES**, a user exit routine must move data into the

variable data area and set the variable data length, or the MOVEVAR control line must move data into the variable data area before processing the ADDDATA control line.

The TSCA contains fields for a pointer to the variable data area (TSCAVDAP) and for the length of that data (TSCAVDAL). The pointer field contains the address of a 512-character variable data area that is allocated when the TSP environment is initialized. Your user exit routine must not modify this pointer.

3. Literal data

Valid reply

A string of 1 to 32 characters of data.

A valid entry in this field indicates that you want the character string added to the data in the command line reply buffer. You enter the data, including any required commas, exactly as you want it added to the command line reply buffer. The data that is entered in this field is collected in the case entered by the user.

Default

No reply.

Restrictions

If the data contains an SBCS comma, you must enter the data in the **Literal data** field, not on the command input line. When an SBCS comma is required as the first or only character of the **Literal data** field, precede the comma with an SBCS space character.

Structured word

If you enter an s-word index when you create the control line, this field is filled in automatically. It displays the actual s-word from the dictionary.

Word acronym

If you enter an s-word index when you create the control line, this field is filled in automatically. It is part of the s-word entry in the dictionary. This field could be blank if a word acronym does not exist for the entry.

Usage Notes and Examples

You can implement several variations to the example on the preceding pages. For instance, suppose you want to create an ADDDATA control line to locate records for people other than Smith. You can define their names in a data set and use the **Get variable data** field in conjunction with a user exit routine. For now, assume that a user exit routine moves a name from the data set to the variable data area and sets the variable data length properly. You now have an ADDDATA control line that contains `6,1,SE + PERA/` in the **Literal data** field. You also set the **Get variable data** field to **YES** to add the name. Remember that the literal data is added to the command line reply buffer first. Then you have another ADDDATA control line that adds the search argument, STAC/OPEN, to the command line reply buffer by using the **Literal data** field. You can insert the space character that separates PERA/name and STAC/OPEN after the name in the user exit routine or before STAC/OPEN in the **Literal data** field.

See “USEREXIT” on page 193 for a discussion of creating user exit routines for a TSP.

Another variation to this example safeguards against someone changing the selection numbers on the Tivoli Information Management for z/OS Primary Options Menu. You want your TSP to work without modification, even after the changes are made. To do this, use the **Structured word index** field to specify the selection from the Primary Options Menu rather than selection number 6. First, create an ADDDATA control line by specifying **002C** (the s-word index for the Inquiry selection) in the **Structured word index** field on the ADDDATA Specification panel.

Next, create an ADDDATA control line that has the search argument *space*`,1,SE + PERA/` in the **Literal data** field. (When an SBCS comma is required as the first, or only, character of the **Literal data** field, you must precede the comma with an SBCS space.) Be sure to set the **Get variable data** field to **YES** to add the name. Then create another ADDDATA control line to add the search argument, **STAC/OPEN**, to the command line reply buffer by using the **Literal data** field. Precede STAC/OPEN with a space to separate PERA/*name* and STAC/OPEN.

You can put up to 512 characters of data in the command line reply buffer. Also, you can enter this data using multiple ADDDATA control lines. However, attempting to store more than 512 characters produces an error condition, and Tivoli Information Management for z/OS does not move any of the data.

For other examples of ADDDATA lines, use PMF to look at TSPs BTNTACLS and BTNTAC1R.

Supplementary Commands for ADDDATA

With the following commands and the control line functions described in this chapter, you can create TSPs that simulate any Tivoli Information Management for z/OS interactive functions:

DOWN LAST

This command scrolls the last data line of a table panel to the top of the screen. The last line becomes the current line of the table panel. For a table panel that contains only display information (such as help panels), the last data line is the line preceding the `BOTTOM OF DATA` line. For all other table panels, this line is the last line on which you can enter data. You can use this command along with the LINECMD command to perform block operations on all records of a search results list.

LINECMD *cmd*

The operand *cmd* can be replaced by any line command that is valid for the current table panel. For example, valid line commands on the Tivoli Information Management for z/OS search results list would be **U, D, P, S,** and **C**. You can specify block line commands when they are enabled. The LINECMD command inserts the operand into the line command area of the current line on the table panel. From that point, normal line command processing occurs. You can use this command to process multiple entries on a table panel. For example, when updating all records on a search results list, enter **linecmd uu**,**down last**,**linecmd uu**. The LINECMD command is not valid when the current panel is not a table panel, or when the table panel does not have a line command area.

By combining the Tivoli-supplied control lines with user exit routines, you can extend the functions of Tivoli Information Management for z/OS to meet the unique requirements of your installation.

What the Control Line Does

When the TSP is run, the data to be added to the command line reply buffer is extracted either from the control line itself or from the variable data area, or from both. The data is added to the command line reply buffer in the TSCA (TSCACMRB) in this order: s-word, literal data, variable data area (depending on which fields you specify in the control line). It is held there until a PROCESS or CLEAR control line is run. The PROCESS control line causes Tivoli Information Management for z/OS to process whatever is in the command line reply buffer. The CLEAR control line causes Tivoli Information Management for z/OS to discard whatever is in the command line reply buffer. In either case, a subsequent ADDDATA control line begins with a clear command line reply buffer. For more information on these control lines, see “PROCESS” on page 152 and “CLEAR” on page 78.

Return and Reason Codes

After an ADDDATA control line is run, the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields are set to indicate what happened. These codes are listed in Table 2.

Table 2. ADDDATA Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion.
8	4	Data movement would cause the command line reply buffer to overflow. The data was not moved. Your TSP has attempted to put more than 512 characters of data into the command line reply buffer since the last PROCESS or CLEAR control line was processed. You cannot construct an IRC greater than 512 characters. To correct the problem, update your TSP to add a PROCESS control line at a convenient point before the ADDDATA control line that produced the unexpected return code.
8	8	Get variable data is specified in the control line but the length field for the variable data area is zero. The data was not moved. Your TSP has not set the length of the variable data field properly. You must set up the variable data area and its length before using an ADDDATA control line that has YES in the **Get variable data** field. The variable data area is set by calling a user exit routine that sets the variable data area length and moves data into it, or by using the MOVEVAR control line. Check your TSP to make sure there is a USEREXIT or MOVEVAR control line in the processing path before the ADDDATA control line that resulted in the unexpected return code. If the TSP has a USEREXIT control line, check the exit routine’s code to make sure it sets the field length properly.
8	12	An internal logic error occurred in National Language Support. Contact your Tivoli representative.

Table 2. ADDDATA Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	16	The user data is not a valid mixed string. Check the data and make the changes required to ensure you specify valid mixed data.

You can use the TESTFIELD control line to test for these return and reason codes. See "TESTFIELD" on page 175 for information on how to do this.

TSCA Field Usage

Tivoli Information Management for z/OS sets the following TSCA fields after an ADDDATA control line is run. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

TSCAFRES
Function reason code

TSCAFRET
Function return code

TSCACMRB
Command line reply buffer

TSCACMOF
Accumulated data length.

If you enter **YES** in the **Get variable data** field and you set the length of the variable data with a user exit routine, the user exit routine must set the length of the variable data in the following TSCA field:

TSCAVDAL
Current user variable data length.

If you set the variable data with the MOVEVAR control line, the MOVEVAR control line sets the field length of TSCAVDAL for you.

ADDLIST

This control line enables you to add one line or a block of lines in a list processor list.

This control line can be called only from a TSX.

The ADDLIST Control Line

The format of the ADDLIST control line is:

```
CALL BLGTSX 'ADDLIST',listsword,stemname,startln,count,panel
```

Parameter Descriptions

1. listsword

Valid reply
The root s-word of the list to which items are to be added. The root s-word includes the hexadecimal watermark character; therefore, it is recommended that you use the

TSX READDICT control line to get the root s-word that you will pass to ADDLIST. Refer to "READDICT" on page 160 for more information on the READDICT control line.

Default

None

Required

2. stemname

Valid reply

The name of a REXX compound variable (including a separator character, such as a period) for the compound variable containing the new items. By convention, the variable name should end with a period. The length is limited to 58 characters, including the period.

Default

BLG_LIST

Optional

3. startln

Valid reply

The line number where the new items should begin. For example, if the value 3 is specified, the items will be inserted after line 2 of the current list. Valid values are 1 to 19274 or LAST (insert the items after the last item in the current list).

Default

LAST

Optional

4. count

Valid reply

The number of items to be added. Valid values are 1 to 19274.

Default

1

Optional

5. panel

Valid reply

The name of the assisted-entry panel or data attribute record used to validate the new items.

Default

None

Required if there are no existing items for the listsword in the record; otherwise, the panel name in the current list is used.

CAUTION:
The list processor data that the TSX ADDLIST control line retrieves must be in the internal format introduced in Version 5.1 (or in Version 4 by APARs OY47188 and OY47893). Lists which were stored prior to the internal format change will not be retrieved accurately by ADDLIST. Those lists can be converted to the new internal format by updating them in Version 7.1, repeating the first line, deleting the first line, and filing the record. The data will be unchanged, but it will be stored in the correct format.

Usage Notes and Examples

This is an example of using an ADDLIST control line in a TSX to add two entries at the end of the list of device names.

```
index='S1416';
CALL BLGTSX 'READDICT',index;   /* Get the s-word.                    */
sword=TSCARSD;                  /* The s-word of the device name list */
newitem.1='Device1'
newitem.2='Device6'
CALL BLGTSX 'ADDLIST',sword,NEWITEM.,'LAST',2
```

This is an example of using an ADDLIST control line in a TSX to insert two items in the list of device names after the third item in the current list.

```
index='S1416';
CALL BLGTSX 'READDICT',index;   /* Get the s-word.                        */
sword=TSCARSD;                  /* The s-word of the device name list     */
newitem.1='Device1'
newitem.2='Device6'
/* Add 2 devices at position 4 (after the third existing item)           */
CALL BLGTSX 'ADDLIST',sword,NEWITEM.,4,2
```

This is an example of using an ADDLIST control line in a TSX to add a blank entry within a list of device names.

```
index='S1416';
CALL BLGTSX 'READDICT',index;   /* Get the s-word.                        */
sword=TSCARSD;                  /* The s-word of the device name list     */
newitem.1='Device1'
newitem.2=' '                   /* A space between the single quotes to signify a blank entry */
newitem.3='Device3'
CALL BLGTSX 'ADDLIST',sword,NEWITEM.,1,3
```

Return and Reason Codes

After the ADDLIST control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 3.

Table 3. ADDLIST Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Items were added successfully; one or more lines existed in the list prior to the add.
0	8	Items were added successfully; one or more blank lines was added between the end of the previous list and the beginning of the new items.
0	12	Items were added successfully; no items existed prior to the add.

Table 3. ADDLIST Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	16	No items existed prior to the add and all listed items were empty rows; therefore, no list was created.
8	4	The function could not be completed because it would cause the list to exceed the maximum allowed size (19274) for a list; items were added up to line 19274.
8	8	The value of a list item failed validation checking. The list is updated and the specified number of lines are added, but the data for the line that failed validation and all subsequent lines is set to blanks. TSCATLIX contains the index of the first item which failed validation.
8	12	An assisted-entry panel name is required when no items exist in the list prior to the add.
8	16	The panel name specified on the control line could not be found.
8	20	The panel name specified on the control line must be an assisted-entry panel or a data attribute record.
8	24	The s-word index specified on the control line does not match the s-word index defined in the panel parameter.
12	4	The control line function failed. This control line cannot be run while the list processor is active.

ADDSDATA

This control line provides the ability to add data to a record. It is the TSX equivalent of the add function of the TSP WORDFIX, described in "Adding Data" on page 210.

Note: The ADDSDATA function should be used only by someone who is very knowledgeable about Tivoli Information Management for z/OS. Care must be taken to ensure that you do not damage your database when using the ADDSDATA function.

This control line can be called only from a TSX.

The ADDSDATA Control Line

The format of the ADDSDATA control line is:

```
CALL BLGTSX 'ADDSDATA',swindex,sword,pfxdata,panel,options
```

Parameter Descriptions

1. swindex

Valid reply

The structured word index for the item to be added (the letter S followed by a four-character EBCDIC value, for example, 'S0BEE').

Default

None

Required if *sword* is specified. If *sword* is not specified, this parameter is not required and anything specified for this parameter is ignored.

2. sword

Valid reply

The structured word for the item to be added. This can be obtained from the dictionary earlier in the TSX using the READDICT control line.

Note: Both the s-word index and the actual s-word must be specified when adding data which contains an s-word. For performance reasons, ADDSDATA does not read the dictionary to retrieve the s-word based on the index. The s-word can be obtained earlier in the TSX using the READDICT control line.

Default

None

Required if *pfxdata* is not specified. If this parameter is not specified, the new data item will not have a structured word.

3. pfxdata

Valid reply

The data (with optional prefix) for the item to be added.

Default

None

This parameter is required if *sword* is not specified. If this parameter is not specified, the new data item will contain no data.

4. panel

Valid reply

The panel name to be associated with the data item to be added.

Default

BLGTXADD.

Optional. If not specified, a name of BLGTXADD will be associated with the item.

5. options

Valid reply

Options to be associated with the data item to be added. Multiple options can be coded, each as a separate parameter. Supported options are:

COGNIZE|COGPWORD|NOCOGNIZE

- **COGNIZE** indicates that both the s-word and p-word are to be cognized.
- **COGPWORD** indicates that only the p-word is to be cognized.
- **NOCOGNIZE** indicates that neither the s-word nor the p-word are to be cognized.

If none of these options is specified, the default is **COGNIZE**. If more than one is specified, the last one is used.

HISTORY|HISTFIRST|NOHISTORY

Additional information about history data is contained in the *Tivoli Information Management for z/OS Panel Modification Facility Guide*, with some detail about the meaning of ORDER and FIRST.

- **HISTORY** indicates that history data for this item is collected in order.
- **HISTFIRST** indicates that history data for this item is to be placed first.
- **NOHISTORY** indicates that no history data is collected.

If none of these options is specified, the default is **HISTORY**. If more than one is specified, the last one is used.

STRING
Indicates that the data portion of the item is string data. If this option is not specified, the data is treated as structured data.

NOPREFIX
Indicates that the data item does not contain a prefix and any slash / or underscore _ characters are part of the data. If this option is not specified and the data item begins with characters that represent a valid prefix, the data item is treated as a prefix and data.

NOREPLACE
Indicates that new data for this field does not replace the previous value, but is added to the previous value. If this option is not specified, the new value replaces the existing value.

COGMIXED
Indicates that the data should be cognized in mixed case. If not specified and the data is being cognized, it is cognized in all uppercase. For most types of data, COGMIXED should not be specified. This option is ignored if NOCOGNIZE is specified.

COGUNPARSED
Indicates that all of the words are cognized as a single string.

Usage Notes and Examples

This is an example of using an ADDSDATA control line in a TSX to add the reporter's phone number, cognize only the p-word, and not keep history data.

```
index='S0B2D';
CALL BLGTSX 'READDICT',index;      /* Get the s-word.                    */
sword=TSCARSD;                     /* The s-word of the reporter's phone */
data='PH/123-4567'
CALL BLGTSX 'ADDSDATA',index,sword,data,,'COGPWORD','NOHISTORY'
```

Note: When using ADDSDATA to add date data, the p-word being added to the record must begin with the characters `DAT`.

Beginning with Version 7.1, you can include nulls in the list of options when invoking ADDSDATA.

```
if fldtype='S' then
    stropt='STRING'
else stropt=''
CALL BLGTSX 'ADDSDATA',index,sword,data,,'NOHISTORY',stropt,'COGPWORD'
```

In previous versions you would have had to code separate calls to ADDSDATA: one with the STRING option and one without the STRING option because the coding in the preceding example would result in an error when `stropt` was null.

Return and Reason Codes

After an ADDSDATA control line is run, the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields are set to indicate what happened. These codes are listed in Table 4.

Table 4. ADDSDATA Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0		Successful completion.
8	4	The requested function was not performed. Your TSX tried to create a data item that was longer than the maximum length accepted by Tivoli Information Management for z/OS. The maximum length for collected data from a response (the s-word, prefixes, and all data associated with those prefixes and control information in the entry) is 256 characters.
8	8	There was not enough storage to process the ADDSDATA control line. Contact your system administrator to increase your region size.
8	16	An internal logic error has occurred. Contact your Tivoli representative.
8	20	An internal logic error occurred in a DBCS function. Contact your Tivoli representative.
8	24	The user data is not a valid mixed string. Check the data and make the changes required to ensure you specify a valid mixed data string.
8	28	The requested data would not be valid Tivoli Information Management for z/OS data. Either the *pfxdata* parameter does not specify any data, or else you specified the REPLACE option but did not specify either a prefix or an s-word.

ADDTEXT

This control line enables you to add lines of new text.

This control line can be called only from a TSX.

The ADDTEXT Control Line

The format of the ADDTEXT control line is:

```
CALL BLGTSX 'ADDTEXT',sword,stemname,startln,count
```

Parameter Descriptions

1. sword

Valid reply

The structured word associated with the text to be added. This can be obtained from the dictionary earlier in the TSX using the READDICT control line.

Default

None

Required

2. stemname

Valid reply

The stem name (including a separator character, such as a period) for the compound variable containing the new items.

Default

BLG_TEXT.

Optional.

3. startln

Valid reply

The line number where the new text should begin. For example, if the value 3 is specified, the items will be inserted after line 2 of the current text. Valid values are 1 to 99999 or LAST (insert the text after the last line of the current text).

Note: A specification of LAST or a specification of a number which is higher than the highest line number in the existing text causes ADDTEXT to append data to the file; a specification of a number which is less than or equal to the highest line number in the existing text causes ADDTEXT to insert data into the file.

Default

LAST

Optional

4. count

Valid reply

The number of lines of text to be added. Valid values are 1 to 99999.

Default

1

Optional

Usage Notes and Examples

This is an example of using an ADDTEXT control line in a TSX to append two lines of text to the bottom of the problem description text.

```
index='S0E01';
CALL BLGTSX 'READDICT',index;   /* Get the s-word.                */
sword=TSCARSD;                  /* The s-word of the text data.   */
newtext.1='First new line'
newtext.2='Second new line'
CALL BLGTSX 'ADDTEXT',sword,'newtext.','LAST',2
```

This is an example of using an ADDTEXT control line in a TSX to insert three lines of text following the first line of problem description text.

```
index='S0E01';
CALL BLGTSX 'READDICT',index;          /* Get the s-word.              */
sword=TSCARSD;                         /* The s-word of the text data. */
newtext.1='First new line'
newtext.2='Second new line'
```

```
newtext.3='Third new line'
/* Add 3 lines starting at line 2 (after existing line 1)              */
CALL BLGTSX 'ADDTEXT',sword,'newtext.',2,3
```

Note: You can use the REXX STRIP() function to remove any trailing blanks that might exist in the text prior to processing the ADDTEXT control line.

Return and Reason Codes

After the ADDTEXT control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 5.

Note: If you are using the Tivoli Information Management for z/OS freeform text editor, you should be aware that this editor adds additional blank lines to a record in order to present a full screen for editing. As a result, the reason code will, in some cases, be different according to whether you do ADDTEXT from within the editor or do ADDTEXT when you are not in the editor.

Table 5. ADDTEXT Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Text was added successfully; one or more lines existed in the text prior to the add.
0	8	Text was added successfully; one or more blank lines were added between the end of the previous text and the beginning of the new text.
0	12	Text was added successfully; no text existed prior to the add.
8	4	The length of one or more input text lines exceeds the maximum allowable line length of 132. No text is updated.

BRANCH

This control line enables you to unconditionally change the flow of the control lines within the TSP.

The function provided by the TSP BRANCH control line is available in a TSX using the standard program flow controls available in the REXX programming language.

You can use this control line to return control from an error-handling portion of the TSP or to simulate a loop when similar processing is done for many records. If you use the BRANCH control line to create a loop, be sure your loop has an ending. You can create the ending by including the target label of a PROCESS, TESTFIELD, or TESTFLOW control line, or with another BRANCH control line that exists outside your loop. This is an example of the BRANCH control line used to simulate a loop.

```
LABEL      TSP
:

LABEL      LOOP
:

TESTFIELD  OUTLOOP       (if TSCAFRET is nonzero, exit the loop)
```

```
PROCESS     ERROR
BRANCH      LOOP
LABEL       OUTLOOP
:
```

Creating a BRANCH Control Line

Use the following BRANCH Specification panel to create a BRANCH control line:

```
BLM8CU9B                  BRANCH  SPECIFICATION                PANEL: ________

Enter 'BRANCH' control data; cursor placement or input line entry allowed.


                  1. Label name.............<R> LOOP____




      When you finish, type END to save or CANCEL to discard any changes.




===>
```

General Rules

The BRANCH control line requires that a target label name, identified with the LABEL control line, exists in the same TSP as the BRANCH control line.

Field Descriptions

1. Label name

Valid reply

A mixed string of 1 to 8 characters, beginning with an SBCS alphanumeric or national character, or an SO character. This field is required.

Default

None.

Restrictions

The label name entered in this field must be a valid name that is identified through the LABEL control line. The LABEL control line cannot immediately precede the BRANCH control line. Also, the name must exist in the same TSP as the BRANCH control line. Tivoli Information Management for z/OS validates this when the TSP is filed.

Usage Notes

The BRANCH control line provides for unconditional branching only. You can use the PROCESS, TESTFIELD, and TESTFLOW control lines for conditional branching.

When you add this control line to a TSP, Tivoli Information Management for z/OS checks for a label name when the TSP is filed. However, no check is made at the time you run the TSP to prevent you from creating an infinite loop. You must verify that the target label name is outside your loop. For examples of TSPs using BRANCH, see TSPs BTNTACLS and BTNTCA21 in PMF.

What the Control Line Does

When the TSP is run, Tivoli Information Management for z/OS resumes processing at the LABEL control line whose name was specified in the BRANCH control line.

Return and Reason Codes

Running a BRANCH control line does not change the TSCA return (TSCAFRET) or reason code (TSCAFRES) fields.

TSCA Field Usage

No TSCA fields are set when the TSP is run.

CLEAR

This control line clears all the data collected in the command-line reply buffer (TSCACMRB) of the TSCA by setting the length of the data stored in the command-line reply buffer to zero (TSCACMOF to X'00000000'). Since the TSP ADDDATA control line stores data in the command-line reply buffer, the TSP CLEAR control line can be used to negate the actions of the TSP ADDDATA control line prior to a TSP PROCESS control line. Once a TSP PROCESS control line processes, the data in the command-line reply buffer is used or is moved to the residual reply buffer (TSCARRB) when the PROCESS encounters an error condition.

Because a TSX accumulates responses to be processed in a REXX variable rather than with ADDDATA control lines, the CLEAR control line is not needed for a TSX. Simply setting the variable to the null string will clear any collected responses.

Assume that you are adding information to the command-line reply buffer. Before the data is processed you want to verify that the current panel name is correct. If it is incorrect, you want your TSP to clear the buffer and then refill the command line reply buffer using an ADDDATA control line. This is an example using the CLEAR control line to clear the command-line reply buffer.

```
    FUNCTION   LABEL   LITERAL                          GET APPLY  FIELD
      NAME     NAME    DATA                             VAR  NOT   NAME
 0. ********* ******** ******************************** **** **** *******
 1. LABEL     TESTTSPP
 2. ADDDATA             6,1,SE + PERA/SMITH              NO
 3. TESTFIELD ERROR     BLG0EN20                         NO   YES  TSCACPNL
 4. PROCESS   EXIT
 5. RETURN
 6. LABEL     ERROR
 7. CLEAR                                                      TSCACMOF
 8. ADDDATA             ;IN,3,2,6,1,SE +PERA/SMITH       NO
 9. PROCESS   EXIT
10. LABEL     EXIT
11. RETURN
```

Creating a CLEAR Control Line

This control line does not collect any input data and, therefore, does not have a specification panel. When you create the CLEAR control line from the Function Name panel, you return to the updated Function Line Summary panel.

Example

For an example of a TSP using the CLEAR control line, use PMF to look at TSP BTNTHS03 in your base panel data set.

What the Control Line Does

When the TSP is run, the accumulated data length for the command-line reply buffer and the return and reason codes are reset to zero. This clears any previous responses from the command-line reply buffer. Processing continues with the next control line in the TSP.

Return and Reason Codes

After a CLEAR control line is run, Tivoli Information Management for z/OS sets the TSCA return (TSCAFRET) and reason (TSCAFRES) code fields to zero. (See Table 6.)

Table 6. CLEAR Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion.

TSCA Field Usage

Tivoli Information Management for z/OS sets the following TSCA fields after a CLEAR is run. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

TSCAFRET
Function return code

TSCAFRES
Function reason code

TSCACMOF
Accumulated data length.

CLOSERRES

This Remote Data Resource TSX control line is described in "CLOSERRES" on page 241.

CLOSESOCKET

This control line closes a TCP/IP socket that had been opened. It uses the assembler callable services to invoke the OS/390 UNIX® System Services version of the TCP/IP product installed. If a callable service does not complete successfully, the name of the service called is returned in the NETFUNC REXX variable, the return code is returned in the NETRETC REXX variable, and the reason code is returned in the NETREAC REXX variable. The user should refer to the OS/390 UNIX System Services Messages and Codes manual and the OS/390 UNIX System Services Assembler Callable Services manual for a description of the returned values.

More than one TCP/IP connection can be established with a TSX concurrently. A socket identifier is returned from OPENSOCKET and is used to specify the connection to be accessed. When the TSX ends, any TCP/IP connections remaining open are closed.

This control line can be called only from a TSX. An ISPF or TSO environment is not required in order to use CLOSESOCKET.

The CLOSESOCKET Control Line

The format of the CLOSESOCKET control line is:

```
CALL BLGTSX 'CLOSESOCKET',socketid
```

Parameter Description

1. socketid

Valid reply

The socket identification for this TCP/IP connection. This value was initially returned from the OPENSOCKET control line in the NETSOCKET REXX variable.

Default

None

Required

Usage Notes and Examples

This is an example of using a CLOSESOCKET control line to close the TCP/IP connection.

```
CALL BLGTSX 'CLOSESOCKET',SaveSocket
```

Note: Six REXX variables have been defined to return information from the CLOSESOCKET control line. These variables (NETSOCKET, NETDATA, NETBYTECOUNT, NETFUNC, NETRETC, and NETREAC) are reset during the processing of CLOSESOCKET. It is the responsibility of the TSX to save any data needed for processing.

Return and Reason Codes

After the CLOSESOCKET control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 7.

Table 7. CLOSESOCKET Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion.

Table 7. CLOSESOCKET Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	4	The TCP/IP service did not complete successfully. Refer to the OS/390 UNIX System Services Messages and Codes manual and the OS/390 UNIX System Services Assembler Callable Services manual for a description of the following return codes. These REXX variables contain the diagnostic information: **NETFUNC** The name of the Assembler Callable Service being invoked. **NETRETC** The value of the Return_code parameter returned by the service. **NETREAC** The value of the Reason_code parameter returned by the service.

DELLIST

This control line enables you to delete one line or a block of lines in a list processor list.

This control line can be called only from a TSX.

The DELLIST Control Line

The format of the DELLIST control line is:

```
CALL BLGTSX 'DELLIST',listsword,startln,count
```

Parameter Descriptions

1. listsword

Valid reply

The root s-word of the list from which items are to be deleted. The root s-word includes the hexadecimal watermark character; therefore, it is recommended that you use the TSX READDICT control line to get the root s-word that you will pass to DELLIST. Refer to "READDICT" on page 160 for more information on the READDICT control line.

Default

None

Required

2. startln

Valid reply

The line number of the first line to be deleted. Valid values are 1 to 19274.

Default

None

Required

3. count

Valid reply
The number of items to be deleted. Valid values are 1 to 19274 or ALL.

Default
1

Optional

CAUTION:
The list processor data that the TSX DELLIST control line retrieves must be in the internal format introduced in Version 5.1 (or in Version 4 by APARs OY47188 and OY47893). Lists which were stored prior to the internal format change will not be retrieved accurately by DELLIST. Those lists can be converted to the new internal format by updating them in Version 7.1, repeating the first line, deleting the first line, and filing the record. The data will be unchanged, but it will be stored in the correct format.

Usage Notes and Examples

This is an example of using a DELLIST control line in a TSX to delete items 8 and 9 in the list of device names.

```
index='S1416';
CALL BLGTSX 'READDICT',index;        /* Get the s-word.                    */
sword=TSCARSD;                       /* The s-word of the device name list */
CALL BLGTSX 'DELLIST',sword,8,2
```

Return and Reason Codes

After the DELLIST control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 8.

Table 8. DELLIST Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	The specified range of list items was deleted successfully.
0	4	The specified range of list items was deleted successfully; the range extended beyond the end of the list.
0	8	The control line completed, but had no effect; no list items exist in the specified range.
0	12	The control line completed, but had no effect; no list items of the specified type exist.
12	4	The control line function failed. This control line cannot be run while the list processor is active.

DELSDATA

This control line provides the ability to delete data from a record. It is the TSX equivalent of the delete function of the TSP WORDFIX, described in “Deleting Data” on page 217, but also provides the ability to delete an entire SDE (s-word XRFI and all p-word XRFIs) with a single control line call.

Notes:

1. The DELSDATA function should be used only by someone who is very knowledgeable about Tivoli Information Management for z/OS. Care must be taken to ensure that you do not damage your database when using the DELSDATA function.
2. DELSDATA does *not* support s-words that use asterisks as a position ignore character. You cannot use an s-word with asterisks on a DELSDATA control line in order to delete multiple s-word entries at one time. You *must* specify one DELSDATA control line for each distinct s-word.

This control line can be called only from a TSX.

The DELSDATA Control Line

The format of the DELSDATA control line is:

```
CALL BLGTSX 'DELSDATA',swindex,sword,pfxdata,panel,options
```

Parameter Descriptions

1. swindex

Valid reply

The structured word index for the item to be deleted (the letter S followed by a four-character EBCDIC value); for example, 'S0BEE'.

Default

None

Conditionally required

If *sword* is specified, a value must also be specified for *swindex*. If *sword* is not specified, any specification for *swindex* is ignored.

2. sword

Valid reply

The structured word for the item to be deleted. This can be obtained from the dictionary earlier in the TSX using the READDICT control line.

Note: Both the s-word index and the actual s-word must be specified when deleting data which contains an s-word. For performance reasons, DELSDATA does not read the dictionary to retrieve the s-word based on the index. The s-word can be obtained earlier in the TSX using the READDICT control line.

Default

None

Conditionally required

Required if *pfxdata* is not specified. If a value is not specified for *sword*, the structured word will not be used to identify the items to delete.

3. pfxdata

Valid reply

The data (with optional prefix) for the item to be deleted.

Default

None

Conditionally required

A value must be specified for *pfxdata* if *sword* is not specified. If no value is specified for *pfxdata*, the data value is not used to identify the items to delete.

4. panel

Valid reply

The panel name to be associated with the data items to be deleted.

Default

None

Optional

If not specified, the panel name is not used to identify the items to delete.

5. options

Valid reply

Options to be associated with locating and deleting the data item. Multiple options can be coded, each as a separate parameter. Supported options are:

STRING

Indicates that *pfxdata* is string data. If STRING is omitted, the data is treated as structured data. If *pfxdata* is not specified, STRING is ignored.

SWORD|DATA|ENTRY

- **SWORD** indicates that only the s-word is to be deleted from the entries, leaving any data. If an entry has no data, the entire entry is deleted.
- **DATA** indicates only the data specified by *pfxdata* is to be deleted, leaving the s-word and any other data. If *pfxdata* is not specified, all data is deleted from the entries, leaving only the s-word. If an entry has no s-word and no data which is not deleted, the entire entry is deleted.
- **ENTRY** indicates that the entire entry, the s-word and all of the data, is to be deleted.

If none of these options is specified, the default is **ENTRY**. If more than one is specified, the last one is used.

Usage Notes and Examples

This is an example of using a DELSDATA control line in a TSX to delete the reporter's phone number (both s-word and p-word).

```
index='S0B2D';
CALL BLGTSX 'READDICT',index;      /* Get the s-word.                    */
sword=TSCARSD;                     /* The s-word of the reporter's phone */
CALL BLGTSX 'DELSDATA',index,sword
```

Return and Reason Codes

After a DELSDATA control line is run, the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields are set to indicate what happened. These codes are listed in Table 9.

Table 9. DELSDATA Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0		Successful completion.

Table 9. DELSDATA Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
4	4	The current record did not have a match for the specified argument. No data was changed. Use the VIEW INTERNALS command to see if what you are trying to locate is in the record. If you are using a p-word search, make sure the argument is complete (prefix/data).
8	8	There was not enough storage to process the DELSDATA control line. Contact your system administrator to increase your region size.
8	20	An internal logic error occurred in a DBCS function. Contact your Tivoli representative.
8	24	The user data is not a valid mixed string. Check the data and make the changes required to ensure you specify a valid mixed data string.

DELTEXT

This control line enables you to delete lines of text.

This control line can be called only from a TSX.

The DELTEXT Control Line

The format of the DELTEXT control line is:

```
CALL BLGTSX 'DELTEXT',sword,startln,count
```

Parameter Descriptions

1. sword

Valid reply

The structured word associated with the text to be deleted.

Default

None

Required

2. startln

Valid reply

The line number of the first line to be deleted. Valid values are 1 to 99999.

Default

None

Required

3. count

Valid reply

The number of lines of text to be deleted. Valid values are 1 to 99999 or ALL.

Default

1

Optional

Usage Notes and Examples

This is an example of using a DELTEXT control line in a TSX to delete lines 10–13 of the problem description text.

```
index='S0E01';
CALL BLGTSX 'READDICT',index;          /* Get the s-word.                */
sword=TSCARSD;                         /* The s-word of the text data.   */
CALL BLGTSX 'DELTEXT',sword,10,4
```

Return and Reason Codes

After the DELTEXT control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 10.

Note: If you are using the Tivoli Information Management for z/OS freeform text editor, you should be aware that this editor adds additional blank lines to a record in order to present a full screen for editing. As a result, the reason code will, in some cases, be different according to whether you do DELTEXT from within the editor or do DELTEXT when you are not in the editor.

Table 10. DELTEXT Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	The specified range of text lines was deleted successfully.
0	4	The specified range of text lines was deleted successfully; the range extended beyond the end of the text.
0	8	The control line completed, but had no effect, because no text exists in the specified range.
0	12	The control line completed, but had no effect, because no text of the specified type exists.

DEQMAIL

This control line is used to dequeue or retrieve a notification mail message from a BLX-SP.

This control line can be called only from a TSX. Also, it is a special purpose control line for handling mail and is not generally useful for writing TSXs.

The DEQMAIL Control Line

The format of the DEQMAIL control line is:

```
CALL BLGTSX 'DEQMAIL',queuename,stemname
```

Parameter Descriptions

1. queuename

Valid reply

The name of the BLX-SP queue from which to retrieve mail.

Refer to "Defining BLX-SP Parameters Members" in the *Tivoli Information Management for z/OS Planning and Installation Guide and Reference* for instructions on how to define the queue name.

Default
MAILQ1

Optional

2. stemname

Valid reply
The name of a REXX compound variable (including a separator character, such as a period) that the freeform mail message will be written to. The number of lines written to the compound variable is contained in *stemname.0*.

Default
BLG_DEQMAIL.

Optional

Usage Notes and Examples

This is an example of using a DEQMAIL control line in a TSX. Mail is read from the default queue and stored in the compound variable DEQMAIL.

```
CALL BLGTSX 'DEQMAIL';         /* Get mail from default queue      */
```

This is another example of using a DEQMAIL control line in a TSX. Mail is read from the default queue and stored in the compound variable MSG.

```
CALL BLGTSX 'DEQMAIL',,'MSG.'  /* Get mail from default queue      */
```

Return and Reason Codes

After the DEQMAIL control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 11.

Table 11. DEQMAIL Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Processing successful. Mail is dequeued from the BLX-SP
0	4	Processing successful; however, the queue is empty.
8	8	Processing unsuccessful. Unable to obtain storage in the user's address space.
12	4	Processing unsuccessful. The specified queue has been closed.
12	8	Processing unsuccessful. The specified queue name does not exist.

FINDSDATA

This control line extracts data that is associated with an s-word or a p-word from the current record, or locates entries associated with a panel. The data being extracted was added by the prompting sequence panels during the record create or update process.

Refer to "The FINDSDATA TSX Control Line" on page 97 for information about using the FINDSDATA control line in a TSX.

This is an example of using the FINDSDATA control line in a TSP that deletes all problems closed during 2000. For this TSP, assume that the database has at least two records that represent closed problems.

Note: Note that in Version 7.1, dates and times are stored in internal format, but are returned to the user in the external format specified by the user. In this example, the DATR/ reflects the internal format. For additional information about BLGEDATE and BLGIDATE, two users exits provided to enable a TSX or TSP to support a variety of external date formats, see "General-purpose User Exits" on page 279.

```
LABEL      DELOLDRC
ADDDATA    3,2,6,1,SE +STAC/CLOSED              (do a search)
PROCESS    ERROR2
ADDDATA    LINECMD SS,DOWN LAST,LINECMD SS      (look at all records)
PROCESS    ERROR2
LABEL      NEXT
TESTFLOW   DONE                      (if at panel BLG1TSRL, then go to
                                      label DONE)
FINDSDATA  DATR/2000.                (look for date)
TESTFIELD  ERROR                     (if TSCAFRET does not equal 0,
                                      go to label ERROR)
ADDDATA    10,4,2,END,END            (delete record)
PROCESS    ERROR
BRANCH     NEXT                      (process next record)
LABEL      ERROR                     (error occurred)
TESTFIELD  ERROR2                    (if TSCAFRES does not equal 4,
                                      go to label ERROR2)
ADDDATA    ;CANCEL                   (end process of record not closed
PROCESS    ERROR2                     during 2000)

BRANCH     NEXT                      (if TSCAFRES equals 4, no data found)
LABEL      DONE
RETURN
LABEL      ERROR2
PRINT
RETURN
```

Creating a FINDSDATA Control Line

You use the following FINDSDATA Specification panel to create a FINDSDATA control line:

```
BLM8CU9F              FINDSDATA SPECIFICATION              PANEL: ________

Enter 'FINDSDATA' control data; cursor placement or input line entry allowed.


 1. Structured word index.... ____          Structured word.. _________
    2. Get list index?....... NO__          Word acronym..... _____________
    3. List index............ 0000
 4. Prefix index............. ____          Prefix.......... ______
 5. Panel name............... ________      Validation....... _______________
 6. Get variable data........ NO_
 7. Word occurrence.......<R> NEXT_
 8. Treat as string data..... NO_
 9. Case-sensitive find?..... NO_

10. Literal data............. DATR/2000._____________________


     When you finish, type END to save or CANCEL to discard any changes.



===>
```

General Rules

You can enter any valid s-word, p-word, or panel name. You can use SBCS and DBCS asterisk (*) and period (.) characters, in accordance with the Tivoli Information Management for z/OS rules, to do a generic search.

Field Descriptions

1. Structured word index

Valid reply

The index key of an s-word in the dictionary data set, or no reply.

An entry in this field causes the FIRST, NEXT, LAST, or PREVIOUS occurrence of the associated s-word in the structured data portion of the current record to be located. You specify the order of location in the **Word occurrence** field described later. Because the watermark characters X'BA', X'BB', X'BC' X'BD', X'BE', or X'BF' that start each s-word cannot be entered, you must use this field to search on an s-word.

Default

No reply.

Restrictions

You cannot specify this field when you enter information in the **Prefix index** field, the **Panel name** field, or the **Literal data** field, or if you enter YES in the **Get variable data** field.

2. Get list index?

Valid reply

YES, NO, or no reply.

This field indicates whether to append the value of TSCA field TSCATLIX to the s-word that corresponds to the value in the **Structured word index** field. When this

field is set to **YES**, any value entered in the **List index** field is assumed to be a hexadecimal value and is not treated as a search operator.

3. List index

Valid reply

A 1- to 2-byte hexadecimal value.

This field indicates that the data to be found was collected with the list processor and that the value in this field should be added to the s-word that corresponds to the value in the **Structured word index** field.

If you specify YES in the **Get list index?** field and specify a value for **List index**, the value in TSCATLIX is added to the value in the **List index** field, and the result is appended to the s-word. The resulting sum is treated as a hexadecimal value and not as a search operator. In other words, if the resulting sum is 5C5C, Tivoli Information Management for z/OS treats this value as a specific value and not **.

If you specify **NO** in the **Get list index?** field and specify a value for **List index**, the **List index** field value is appended to the s-word, and the value of TSCATLIX is ignored. If the **List index** field contains a X'5C' or X'4B' (SBCS asterisk and period respectively) then these are treated as search operators. If X'4Bxx' is entered, then the last byte is ignored (*xx* is any valid hexadecimal value).

4. Prefix index

Valid reply

The index key of a p-word in the dictionary data set or no reply.

If you specify this field by itself, you must refer to a p-word that has a literal validation pattern and possibly a prefix. The validation pattern must be enclosed in delimiter symbols (<>), indicating literal data. You can also specify a prefix index that refers to a prefix without any associated validation pattern. When you do this, you must also enter **YES** in the **Get variable data** field to have variable data from the TSCA added after the prefix, or you can specify a value in the **Literal data** field to be added after the prefix.

An entry in this field causes the FIRST, NEXT, PREVIOUS, or LAST occurrence of the complete p-word/data combination in the structured data portion of the current record to be located. You specify the order of location in the **Word occurrence** field, described below.

Default

No reply.

Restrictions

A prefix that has a nonliteral data validation pattern (such as AAV3 or <S>NR4) is not valid. If you use the **Prefix index** field and the associated p-word does not contain a validation pattern, then you must either enter **YES** in the **Get variable data** field or enter data in the **Literal data** field. You cannot use this field if you entered data in the **Structured word index** field or in the **Panel name** field.

5. Panel name

Valid reply

The name of a panel to be located in the structured data portion of the record or no reply.

If you specify this field, the FIRST, NEXT, LAST, or PREVIOUS occurrence of the panel name in the structured data portion of the current record is located. You specify the order of location in the **Word occurrence** field, described later.

Default

No reply.

Restrictions

You cannot use this field if you entered data in the **Structured word index** field, the **Prefix index** field, or the **Literal data** field, or if you entered YES in the **Get variable data** field.

6. Get variable data

Valid reply

YES, NO, or no reply.

When you enter **NO** or make no reply, this field has no effect. When you enter **YES** it indicates that you want data in the variable data area to be added to a prefix, if one was specified, or that the variable data area contains the entire search argument.

Default

NO

Restrictions

If you specify an index for a prefix without an associated validation pattern, you must either enter **YES** in this field or fill in the **Literal data** field. You cannot use this field if you entered data in the **Structured word index** field, the **Panel name** field, or the **Literal data** field.

If the FINDSDATA control line has **YES** in this field, a user exit routine must move data into the variable data area and set the variable data length, or the MOVEVAR control line must move data into the variable area before the FINDSDATA control line is processed. The TSCA contains fields for a pointer to the variable data area (TSCAVDAP) and for the length of that data (TSCAVDAL). The pointer field contains the address of a variable data area that is allocated when the TSP environment is initialized. Your exit routine must not modify this pointer.

7. Word occurrence

This field indicates how to search for an entry in the structured data portion of the current record. An entry is usually located through a search word, which can be specified in the **Structured word index**, **Prefix index**, **Literal data**, or **Panel name** field, or as variable data. This field is required.

Valid reply

A reply of FIRST indicates that the first occurrence of the search word is to be located. If no search word is specified, the first data item in the record is returned.

A reply of NEXT indicates that the next data item or occurrence of the search word is to be located. The search starts at the entry following the data item last found. An internal pointer to a current data item is maintained. Each time a FINDSDATA operation is performed, this pointer is updated to point to the item that was located.

The first time the FINDSDATA function is used for a record, NEXT causes the search to start with the first data item in the record. If no search word is specified,

the next data item is returned. When a PROCESS control line is run, the internal pointer is reset. NEXT, specified without a search argument, is a good way of stepping through the entries of a record.

When coding multiple FINDSDATA statements in a TSP without resetting the current line pointer, either always specify a search argument or never specify a search argument. Results are unpredictable if you use FINDSDATA NEXT without a search argument after a FINDSDATA with a search argument.

A reply of LAST indicates that the last occurrence of the search word is to be located. If no search word is specified, the last data item in the record is returned. Use LAST without a search word to find data most recently added to a record. Use LAST with a search word to find the most recent occurrence of that search word in a record.

A reply of PREV indicates that the previous occurrence of the search word is to be located. This allows a backwards search. If you specify no search word, the previous data item is returned.

Default

NEXT

Restrictions

None.

8. Treat as string data

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that the control line should treat the search word as string data so only string data fields in the record are found. A string data field is treated as a single-character string during data-entry, rather than as multiple words, and can therefore contain special characters.

For example, the **Description abstract** field of a problem record can contain several words. However, when this field is designated as a string data field (that is, the **Collect as string** field on the associated assisted-entry panel is set to **YES**), the entire field, including words and spaces between, is treated as one character string.

If you use **YES**, the entire string must be specified. If you are using non-Latin translation tables, no global characters (* .) can be used to search for string data.

If you use **NO**, only nonstring data is searched for.

Default

NO

Restrictions

None.

9. Case-sensitive find?

Valid reply

YES, NO, or no reply.

A reply of **YES** in this field indicates that you want the character string specified in the **Literal data** field to be used as a case-sensitive argument. A reply of **NO** in this

field indicates that the case of a character should be ignored when locating data which matches the value specified in the **Literal data** field.

Default

NO

Restriction

The capability of using case-sensitive input data depends on whether the Latin translate table **BLGPTRTB** or the non-Latin translate table **BLGPTRTK** was specified when Tivoli Information Management for z/OS was installed. For more information on translate tables, refer to the *Tivoli Information Management for z/OS Planning and Installation Guide and Reference*.

10. Literal data

Valid reply

A string of 1 to 32 characters of data, or no reply.

A valid entry in this field indicates that you want the character string to be added to a prefix specified in the **Prefix index** field. If no prefix index is specified, the literal data is treated as the actual search word. If you specified **YES** in the **Case-sensitive find?** field above, the search argument that you enter here will be searched sensitive to the case in which you entered the data.

Default

No reply.

Restriction

If you specify a **Prefix index** field that does not contain an associated validation pattern, you must either enter YES in the **Get variable data** field or enter data in this field. You cannot specify this field if data is entered in the **Structured word index** field or the **Panel name** field, or if you enter **YES** in the **Get variable data** field.

Structured word

If you enter an s-word index when you create the control line, this field is filled in automatically. It displays the actual s-word found in the dictionary.

Word acronym

If you enter an s-word index when you create the control line, this field is filled in automatically. It is part of the s-word entry in the dictionary. This field could be blank if a word acronym does not exist for the entry.

Prefix

If you enter a prefix index when you create the control line, this field is filled in automatically. It displays the actual p-word found in the dictionary. Check this entry to make sure it is the information you expect to locate. If the dictionary entry was changed between the time that the original data was collected and the time that the TSP was written, Tivoli Information Management for z/OS might not find the expected information.

Validation

If you enter a prefix index when you create the control line, this field is filled in automatically. It displays the validation pattern associated with the prefix index.

Usage Notes and Examples

The results of a FINDSDATA search are returned in the same sequence as shown in a View Internal Data panel. Figure 1 shows an example for a problem record that is being created.

```
BLG1TVID                        VIEW INTERNAL DATA                    LINE 1 OF 9

PANEL    PANEL  REL COG-  FLAGS    SWORD  STRUCTURED    PREFIX WORD OR
NAME    TYP/RSP LEV NIZE F  M  D   INDEX     WORD       VISIBLE PHRASE

BLG0EN20  D/ 5  06  B/   18/00/88  S002C   BA //S/TSI   ENTRY
BLG00000  S/ 1  06  B/   18/04/08  S0032   BA //S/TXS   RECS=PROBLEM
BLG0B001  S/ 1  06  B/   19/00/80  S0CFC   BC IMDIAENT0 REPORTER
BLG0B100  D/ 1  06  N/   41/00/08  S0B59
BLG6REQN  A/ 2  06  B/U  0D/04/00  S0B59   BC IM0I0PS00 PERS/POWELL
BLG0B100  D/14  06  N/   41/00/00  S0BEE
BLG6STAT  A/ 2  06  B/U  0D/04/00  S0BEE   BC IMS0SSC00 STAC/OPEN
BLG0B100  D/25  06  N/   41/00/00  S0E0F
BLG6DSAB  A/ 2  06  B/   0D/04/00  S0E0F   BC IM0TXCA00 TEST PROBLEM
*** BOTTOM OF DATA ***



 Type DOWN or UP to scroll the panel, or type END to exit the panel.

===>
```

Figure 1. View Internal Data Panel Showing a Record That Has Never Been Filed. The data listed under the headings PANEL NAME, STRUCTURED WORD, and PREFIX WORD OR VISIBLE PHRASE is what FINDSDATA examines.

If you want to find the problem type for this record, you must use FINDSDATA specifying LAST with the TYPE/. search word to find the current problem type value. Using FINDSDATA specifying TYPE/SOFTWARE does not give valid results in this example because the record type was changed from software to hardware, as shown on the last line of the View Internal Data panel. Similarly, specifying FIRST with a search word of TYPE/. also yields results that are not valid. LAST gives a valid result in this example because it locates the most recent occurrence of the search word.

Figure 2 on page 95 shows an example of View Internal Data panels for a previously created problem record.

```
BLG1TVID                      VIEW INTERNAL DATA                 LINE 1 OF 24

PANEL    PANEL  REL COG-  FLAGS    SWORD  STRUCTURED    PREFIX WORD OR
NAME    TYP/RSP LEV NIZE F  M  D   INDEX     WORD       VISIBLE PHRASE

BLG0EN20  D/ 5  06  B/   18/00/88  S002C   BA //S/TSI   ENTRY
SDDSROOT  C/    00  N/   12/04/10  S0000                00000EBE
BLG00000  S/ 1  06  B/   18/04/08  S0032   BA //S/TXS   RECS=PROBLEM
BLG0B001  S/ 1  06  B/   19/00/00  S0CFC   BC IMDIAENT0 REPORTER
BLG0B100  D/ 1  06  N/   41/00/08  S0B59
BLG6REQN  A/ 2  06  B/U  0D/04/00  S0B59   BC IM0I0PS00 PERS/POWELL
BLG0B100  D/14  06  N/   41/00/00  S0BEE
BLG6STAT  A/ 2  06  B/U  0D/04/00  S0BEE   BC IMS0SSC00 STAC/OPEN
BLG0B100  D/25  06  N/   41/00/00  S0E0F
BLG6DSAB  A/ 2  06  B/   0D/04/00  S0E0F   BC IM0TXCA00 TEST PROBLEM
BLG0B100  D/ E  06  N/   01/00/0C  S000B
BLGSDBCP  D/ 9  06  N/   19/00/5C  S0CF1   BC ISSCPN009 File record
BLG1A111  C/ 2  06  B/U  0C/00/00  S0C34   BC IM00SDC00 DATE/05/21/2000 DATE/05
                                                       21/2000
BLG1A111  C/ 2  06  B/U  0C/00/00  S0C61   BC IM00STC00 TIME/17:52 TIME/17:52


===>
```

```
BLG1TVID                      VIEW INTERNAL DATA                LINE 15 OF 24

PANEL    PANEL  REL COG-  FLAGS    SWORD  STRUCTURED    PREFIX WORD OR
NAME    TYP/RSP LEV NIZE F  M  D   INDEX     WORD       VISIBLE PHRASE

BLG1A111  C/ 2  06  B/U  0C/00/00  S0BB1   BC IM0I0CCS0 CLAE/MASTER
BLG1A111  C/ 2  06  B/U  0D/00/00  S0C35   BC IM00SDM00 DATM/05/21/2000 DATM/05
                                                       21/2000
BLG1A111  C/ 2  06  B/U  0D/00/00  S0C62   BC IM00STM00 TIMM/17:52 TIMM/17:52
BLG1A111  C/ 2  06  B/U  0D/00/00  S0B5E   BC IM0I0PM00 USER/GSWOOD
BLG1A111  C/ 2  06  B/U  0D/00/00  S0D90   BC PROBN0D90 ESCL/1
BLG1A115  C/ 2  00  N/   0D/04/10  S0000   BC ISRAC3010 5
BLGCFILE  S/    00  B/   99/04/10  S0000   BC ISSCPR002 MANAGEMENT
BLGCFILE  C/ 2  00  N/   8D/04/10  S0000   BC IM00SST00 AEB1886271D72215
BLGCFILE  C/ 2  00  N/   8D/04/10  S0000   BC IM00SST01 AEB1886271D72215
BLGCADRN  A/ 2  00  B/U  8D/04/10  S0000   BC IM00NR001 RNID/00000008
*** BOTTOM OF DATA ***


 Type DOWN or UP to scroll the panel, or type END to exit the panel.

===>
```

Figure 2. View Internal Data Panels Showing a Record That Was Filed Previously

If you want to step through the structured description entries of a record, use FINDSDATA specifying FIRST as a parameter with no search argument, followed by FINDSDATA NEXT in a loop. Using the above problem record, the data returned in the TSCA is as follows:

FIRST

S-Word X'BA' //S/TSI in TSCARSD. Visible phrase ENTRY is in TSCAVPH

NEXT

S-Word X'BA' //S/TXS in TSCARSD. Visible phrase RECS=PROBLEM is in TSCAVPH

NEXT

S-Word X'BC' IMDIAENT0 in TSCARSD. Visible phrase REPORTER is in TSCAVPH

NEXT
TSCAFRES=4, because there is no data associated with this entry.

⋮

NEXT
Return code 8 in TSCAFRET and reason code 4 in TSCAFRES, indicating the end of the record.

Using this same record, if you request the last occurrence of panel BLG0B100, Tivoli Information Management for z/OS sets the return code field to 0 and the reason code field to 0, indicating that it found the panel. Follow this control line with another FINDSDATA specifying the NEXT parameter with no search argument. Tivoli Information Management for z/OS returns the s-word, X'BC' ISSCPN009, in TSCARSD and the visible phrase, `File record`, in TSCAVPH.

Another example using this record is to request the first occurrence of the prefix CLAE/. in the data. Tivoli Information Management for z/OS returns both the s-word and p-word for that entry and sets the current position to that entry. Now you want to locate the first occurrence of the prefix DATE/. in the data.

If you request a FINDSDATA FIRST or FINDSDATA LAST for DATE/., Tivoli Information Management for z/OS finds it because a request of FIRST resets the current position to the top of the record. If you request a FINDSDATA NEXT for DATE/. after having located the prefix CLAE/., Tivoli Information Management for z/OS does not find it, because the DATE/. entry precedes the current position. For more examples, use PMF to look at TSPs BTNTAPRV and BTNTAC1R in the base panel data set.

If you want to code multiple FINDSDATA control lines in a TSP, you should either always or never specify a search argument. Do not code some FINDSDATA control lines with a search argument and some without, or you will get unpredictable results.

What the Control Line Does

The first FINDSDATA request establishes the current position in the record. The current position is updated for each successful FINDSDATA request until a PROCESS control line is found, at which point the current position is cleared. If a FINDSDATA request fails, the current position is not updated. The first FINDSDATA request that occurs after a PROCESS control line starts from the beginning of the current record, and the current position is reestablished.

When a FINDSDATA runs successfully, the data (if any) is stored in fields in the TSCA.

- If you specify a search by s-word index, the following fields are set:
  TSCARSD
  The located s-word
  TSCARSDL
  The length of the located s-word
  TSCARPD
  If any, the prefix associated with the found item
  TSCARPDL
  The length of the associated prefix
  TSCASDF
  If any, the validation data associated with the found item

TSCASDFL
The length of the associated data
TSCAVPH
If any, the visible phrase associated with the found item
TSCAVPHL
The length of the visible phrase.

- If you specify a search by p-word, the following fields are set:
TSCARPD
The located prefix
TSCARPDL
The length of the located prefix
TSCASDF
The validation data associated with the found item
TSCASDFL
The length of the associated data
TSCARSD
If any, the s-word associated with the found item
TSCARSDL
The length of the associated s-word.

- If you specify a panel, a position is established in the current record for that data item, but no data is returned.

The FINDSDATA TSX Control Line

The format of the FINDSDATA control line is:

```
CALL BLGTSX 'FINDSDATA',searchword,searchtype,occurrence,date/time form
```

Parameter Descriptions

1. searchword

Valid reply

The string to be searched for. Valid values for searchword vary based on the search type.

DATA 1 to 256 characters

CSDATA
1 to 256 characters

PANEL
8 characters (panel name)

STRING
1 to 256 characters

CSSTRING
1 to 256 characters

SWORD
2 to 10 characters, with the first between X'BA' and X'BF'. The TSX READDICT control line can be used to get the s-word from the dictionary. Refer to “READDICT” on page 160 for more information on the READDICT control line.

Note: Refer to “GETLIST” on page 119 for information on how to retrieve list processor data from a record using a TSX.

Default

None

If omitted, *any* item will match the search.

Optional

2. searchtype

Valid reply

The type of search.

DATA Search for a data field containing the indicated prefix/data or unprefixed data.

Note: The DATA searchtype can also be used to locate the user's local form of a date.

CSDATA

A case-sensitive search for a data field containing the indicated prefix/data or unprefixed data.

PANEL

Search for data entries collected from the specified panel name.

STRING

Search for a string data field with the specified data (equivalent to DATA, but with the ″Treat as string data″ field specified in a TSP set to YES instead of NO).

CSSTRING

A case-sensitive search for a string data field with the specified data. In the TSP version of FINDSDATA, there is a TREAT AS STRING DATA field (on page 92) with which you can indicate that the data you are looking for is string data. CSSTRING performs a find like you would get in a TSP if you specified both TREAT AS STRING DATA=YES and CASE SENSITIVE FIND?=YES in the TSP.

SWORD

Search for a data field with the specified s-word.

UT Search for a Universal Time date or time argument. When UT is specified, only the universal time value for each field is checked for a matching value.

Note: Universal time values are in internal format: YYYY/MM/DD.

OLOCAL

Search for a Local Time date or time argument. When OLOCAL is specified, only the original local time value (the local time of the user who last changed the data) for each field is checked for a matching value.

Note: Local time values are in internal format: YYYY/MM/DD.

Default

DATA

Optional

3. occurrence

Valid reply
Which item to search for.

FIRST
The first item in the record containing the searchword.

LAST The last item in the record containing the searchword.

NEXT
The next item (following the item last found) containing the searchword.

PREV The previous item (before the item last found) containing the searchword.

Default
NEXT

Optional

4. date/time form

Valid reply
Which item to search for.

UT Return the universal time date or time (internal format).

OLOCAL
Return the original local date or time (internal format).

ULOCAL
Return the date or time in the current user's local time zone and external date format.

Default
ULOCAL

Optional

Usage Notes and Examples

This is an example of using a FINDSDATA control line in a TSX. The first occurrence of the assignee name is located in the record.

```
CALL BLGTSX 'FINDSDATA','PERA/.','DATA','FIRST';
```

The arguments used to locate data must be in the proper format for the selected form. For example, FINDSDATA control lines with ULOCAL of DATA must specify the user's local form of the date or time. If the user's date format is DD-MM-YYYY and you want to perform a FINDSDATA for any dates occurring in May 2000, you must use an argument of DATO/**-05–2000. A FINDSDATA specifying the ULOCAL or DATA searchtype will not find any UT or original local date values, even if the argument matches the data. UT or OLOCAL arguments must be in internal format (YYYY/MM/DD or HH:MM).

Return and Reason Codes

After a FINDSDATA control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 12 on page 100.

Table 12. FINDSDATA Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion.
0	4	The control line ran successfully, but no structured data exists for this entry.
0	8	Data was found, but it is flagged for deletion when the current record is filed. The field was filled in and then blanked out. If the found item is a list processor item, then the item is eliminated when the current list processor call ends. This return and reason code combination applies only to data marked for deletion, not to data that will be replaced later by data with the same p-word, s-word, or both.
0	12	List item data was found. TSCATLIX is updated with the index.
8	4	The requested item was not found. If the data is supposed to be in the record, verify that your TSP is building a correct search argument. You can also view internals to see if what you are searching for is in the record. The current position might be past the data because of a previous FINDSDATA. When you use the FIRST parameter, the internal pointer is reset to the beginning of the record.
8	8	Get variable data is specified in the control line, but the length field for the variable data area is zero. Your TSP must set up the variable data area and its length before the FINDSDATA control line can be run. You can either call a user exit routine that sets the variable data area length and moves data into it, or use the MOVEVAR control line. Check your TSP to make sure a USEREXIT or MOVEVAR control line appears in the processing path before the FINDSDATA control line that resulted in the unexpected return code. If USEREXIT control line is in the TSP, check the exit routine's code to make sure that it sets the length field properly.
8	10	The sum of the lengths of the list index and the s-word to be searched on exceeds the length of the s-word field in the TSCA.
8	12	An internal logic error occurred. Contact your Tivoli representative.
8	16	The data being used to build the search argument is too long. The maximum length of the search argument is 252 characters. To correct the problem, either update your TSP with a search argument that is no more than 252 characters long, or check to make sure that your variable data does not cause the search argument to be exceeded.
8	20	The sum of the lengths of the list index and the s-word to be searched on exceeds the length of the s-word field in the TSCA.

Table 12. FINDSDATA Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	24	An internal logic error occurred in National Language Support. Contact your Tivoli representative.
8	28	The user data is not a valid mixed string. Check the data and make the changes required to ensure you specify valid mixed data.

You can use the TESTFIELD control line to test for these return and reason codes. See "TESTFIELD" on page 175 for information on how to do this.

TSCA Field Usage

Tivoli Information Management for z/OS sets the following TSCA fields after a FINDSDATA control line is run. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

TSCAFRET
: Function return code

TSCAFRES
: Function reason code

TSCAVPHL
: Visible phrase length

TSCAVPH
: Visible phrase

TSCARSDL
: S-Word length

TSCARSD
: S-Word

TSCARPDL
: P-Word length

TSCARPD
: P-Word

TSCASDFL
: Data field length

TSCASDF
: Structured data field

TSCATLIX
: List index field.

In a TSP if you enter **YES** in the **Get variable data** field and you set the variable data with a user exit routine, the user exit routine must set the length of the variable data in the following TSCA field:

TSCAVDAL
: Current user variable data length

If you set the variable data with the TSP MOVEVAR control line, the MOVEVAR control line sets field TSCAVDAL for you.

FINDSJRNL

This control line extracts history data associated with an s-word or a p-word from the journal portion of the current record.

Refer to "The FINDSJRNL TSX Control Line" on page 107 for information about using the FINDSJRNL control line in a TSX.

Creating a FINDSJRNL Control Line

Use the following FINDSJRNL Specification panel to create a FINDSJRNL control line:

```
BLM8CU9E               FINDSJRNL SPECIFICATION               PANEL: ________

Enter 'FINDSJRNL' control data; cursor placement or input line entry allowed.


 1. Structured word index.... ____            Structured word.. _______________
 2. Prefix index............. ____            Word acronym..... _______________
 3. Get variable data........ NO_             Prefix.......... ______
 4. Data occurrence.......<R> OLDER_          Validation....... _______________
 5. Case-sensitive find?..... NO_

 6. Literal data............. DATM/._____________________________




       When you finish, type END to save or CANCEL to discard any changes.




  ===>
```

General Rules

The specified s-word or p-word can identify data entered during the record create or update process or control data added by the system. You can use SBCS and DBCS asterisk (*) and period (.) characters to do a generic search, in accordance with the *Tivoli Information Management for z/OS User's Guide* description of Tivoli Information Management for z/OS rules.

Field Descriptions

1. Structured word index

Valid reply

The index key of an s-word in the dictionary or no reply.

An entry in this field causes the NEWEST, OLDER, or OLDEST occurrence of the associated s-word to be located in the journal portion of the current record. You specify the order of location in the **Word occurrence** field.

Default

No reply.

Restrictions

You cannot specify this field if you enter information in the **Prefix index** field or the **Literal data** field, or if you enter **YES** in the **Get variable data** field.

2. Prefix index

Valid reply

The index key of a prefix in the dictionary data set or no reply.

If you specify this field only, you must refer to a p-word that has a literal validation pattern and possibly a prefix. The validation pattern must be enclosed in delimiter symbols (<>), indicating literal data. You can specify a prefix index that refers to a prefix without any associated validation pattern. When you do this, either you must specify the **Get variable data** field as **YES**, so variable data from the TSCA is added after the prefix, or you must specify after the prefix a value to be added in the **Literal data** field. An entry in this field causes the NEWEST, OLDER, or OLDEST occurrence of the complete prefix and data combination to be located in the journal portion of the current record. You specify the order of location in the **Data occurrence** field.

Default

No reply.

Restrictions

A nonliteral data validation pattern (such as AAV3 or <S>NR4) is not valid. If you use the **Prefix index** field and the associated p-word does not contain a validation pattern, you must either enter **YES** in the **Get variable data** field or enter data in the **Literal data** field. You cannot use this field if you entered data in the **Structured word index** field.

3. Get variable data

Valid reply

YES, NO, or no reply.

When you enter **NO** or make no reply, this field has no effect. When you enter **YES**, it indicates that you want data in the variable data area to be added to a prefix (if one was specified) or that the variable data area contains the entire search argument.

Default

NO

Restrictions

If you specify a prefix index for a prefix without an associated validation pattern, you must either enter **YES** in this field or fill in the **Literal data** field. You cannot use this field if you enter data in the **Structured word index** field or the **Literal data** field.

If the FINDSJRNL control line has **YES** in this field, a user exit routine must move data into the variable data area and set the variable data length, or the MOVEVAR control line must move data into the variable data area before processing the FINDSJRNL control line. The TSCA contains fields for a pointer to the variable data area (TSCAVDAP) and for the length of that data (TSCAVDAL). The pointer field contains the address of a variable data area that is allocated when the TSP environment is initialized. Your exit routine must not modify this pointer.

4. Data occurrence

This field indicates how to search for an entry in the journal portion of the record. An entry is usually located through a search word, which can be specified in the **Structured word index** field, **Prefix index** field, **Literal data** field, or as variable data. This field is required.

Valid replies

A reply of NEWEST indicates that the most recent occurrence of the search word is to be located. If no search word is specified, the newest entry in the journal is returned.

A reply of OLDER indicates that an older occurrence of the search word is to be located. The search starts at the next older entry following the data item last found. An internal pointer to a current journal entry is maintained. Each time a FINDSJRNL operation is performed, this pointer is updated to point to the item located. The first time the FINDSJRNL control line is used for a record, OLDER causes the search to start with the first data item in the record. If no search word is specified, the newest entry in the journal is returned. Whenever a PROCESS control line is run, the internal pointer is reset. OLDER, specified without a search argument, is a good way of stepping through the entries in the journal.

A reply of OLDEST indicates that the oldest occurrence of the search word is to be located. If no search word is specified, the oldest entry in the journal is returned.

Default

OLDER

Restrictions

None.

5. Case-sensitive find?

Valid reply

YES, NO, or no reply.

A reply of **YES** in this field indicates that you want the character string specified in the **Literal data** field to be used as a case-sensitive argument. A reply of **NO** in this field indicates that the case of a character should be ignored when locating data which matches the value specified in the **Literal data** field.

Default

NO

Restriction

The capability of using case-sensitive input data is dependent on whether the Latin translate table or the non-Latin translate table was specified when Tivoli Information Management for z/OS was installed. The *Tivoli Information Management for z/OS Planning and Installation Guide and Reference* contains additional information regarding translate tables.

6. Literal data

Valid reply

This depends upon how your SDIDS key length is set:

- A string of 1 to 32 characters or no reply. However, if the prefix index field is filled in, the sum of the bytes in the prefix index field and in the literal data field cannot exceed 32 bytes.

Note: The length of the character string is not dependent on the SDIDS key length.

A valid entry in this field indicates that you want the character string to be added to a prefix specified in the **Prefix index** field. If no prefix index is specified, the literal data is treated as the actual search word. If you specified **YES** in the CASE-SENSITIVE FIND? field above, the search argument that you enter here will be searched sensitive to the case in which you entered the data.

Default

No reply.

Restrictions

If you specify a **Prefix index** field that does not contain an associated validation pattern, you must enter **YES** in the **Get variable data** field, or enter data in this field. You cannot specify this field if the **Structured word index** field has data or if you enter **YES** in the **Get variable data** field.

When an SBCS comma is required as the first, or only, character of the **Literal data** field, you must precede the comma with an SBCS space character.

Structured word

If you enter an s-word index when you create the control line, this field is filled in automatically. It displays the actual s-word found in the dictionary.

Word acronym

If you enter an s-word index. when you create the control line, this field is filled in automatically. It is part of the s-word entry in the dictionary. This field could be blank if a word acronym does not exist for the entry.

Prefix

If you enter a prefix index when you create the control line, this field is filled in automatically. It displays the actual prefix found in the dictionary. Check this entry to make sure it is the information you expect to locate. If the dictionary entry was changed between the time that the original data was collected and the time that the TSP was written, Tivoli Information Management for z/OS might not find the expected information.

Validation

If you enter a prefix index when you create the control line, this field is filled in automatically. It displays the validation pattern associated with the prefix index.

Usage Notes

The FINDSJRNL control line examines the journal data in a record. A journal entry is created each time you update a history (<H>) field in a record. The new data and control information are recorded in the journal entry. To help you better understand how FINDSJRNL works, look at the History Display panel shown in Figure 3 on page 106.

```
BLG1TDHD                      HISTORY DISPLAY            LINE 1 OF 9

 DATE         TIME      USERID     JOURNALIZED HISTORY DATA
ALTERED      ALTERED

04/04/2000    13:26     B123456    CLAO/PSMITH   DATA/04/04/2000 GROA/ABC
                                    GROC/DEF   MISB/NO   NASY/SYSTEM
                                    PERA/SMITH   PERC/JONES   PRII/2
                                    PRIO/2   STAC/OPEN   TIMA/16:05
04/17/2000    15:16               CODC/PERE
04/18/2000    07:20     B789543    CLAT/CJONES   PERA/CJONES
              08:04               CLAO/CJONES   CODC/PERN
              09:20
04/19/2000    09:11     B123456    STAC/CLOSED









===>
```

Figure 3. History Display Panel Example

If you want to step through the entries in the journal, you can use FINDSJRNL, specifying NEWEST as a parameter with no search argument, followed by FINDSJRNL OLDER in a loop. Using the history record at the beginning of this section, the data returned in the TSCA is as follows:

NEWEST

S-Word associated with STAC/CLOSED in TSCARSD
P-Word STAC/ in TSCARPD and structured data CLOSED in TSCASDF

OLDER

S-Word associated with USER/B123456 in TSCARSD
P-Word USER/ in TSCARPD and structured data B123456 in TSCARSD

OLDER

S-Word associated with TIMM/09:11 in TSCARSD
P-Word TIMM/ in TSCARPD and structured data 09:11 in TSCASDF

OLDER

S-Word associated with DATM/04/19/2000 in TSCARSD
P-Word DATM/ in TSCARPD and structured data 04/19/2000 in TSCASDF
.
.
.

OLDER

P-Word DATM in TSCARPD and structured data 04/04/2000 in TSCASDF

OLDER

Return code 8 in TSCAFRET and reason code 4 in TSCAFRES, indicating end of the journaled data.

You can use the FINDSJRNL control line to locate the dates when the status was changed by first using FINDSJRNL NEWEST with a search argument of STAC/. to find the most recent status entry. FINDSJRNL OLDER with a search argument of DATM/. locates the date of the status entry. Using the two FINDSJRNL OLDER control lines to alternately search on STAC/. and DATM/., you can step through the record until all occurrences are found.

Note: If Universal Time processing has been enabled for your application, a **Date Modified** history entry is only built if the local date of the user making the change is different than it was for the previous change. Therefore, a U.S. Pacific Time user who makes a change at 18:00 on 2/20/01 and another change at 23:00 the same night will not have a second **Date Modified** entry generated for the second change. However, to a U.S. Eastern Time user, the history data for the Pacific Time user's changes will appear as follows:

```
02/20/2001  21:00
            02:00
```

The date for the second change, when viewed in U.S. Eastern Time, should be 02/21/2001; but because the Pacific Time user's date did not turn between changes, the Eastern Time user's view does not display a date change either. However, because history entries are always listed in chronological order, you can tell when a date change should occur when viewing histories of records originating in another time zone.

What the Control Line Does

The first FINDSJRNL request establishes the current position in the record. The current position is updated for each additional FINDSJRNL request until a PROCESS control line is found, at which point the current position is cleared. If a FINDSJRNL request fails, the current position is not updated. The first FINDSJRNL request that occurs after a PROCESS control line starts at the newest journaled data of the current record, and the current position is reestablished.

When a FINDSJRNL control line is run successfully, the data, if any, is stored in fields in the TSCA as follows:

- If you specify a search for an s-word, the s-word data is stored in TSCARSD and TSCARSDL and the visible phrase is stored in TSCAVPH and TSCAVPHL.
- If you specify a p-word search, the prefix is stored in TSCARPD and TSCARPDL and the data associated with the prefix is stored in TSCASDF and TSCASDFL. If an s-word is associated with the found item, its data is stored in the fields noted earlier.

The FINDSJRNL TSX Control Line

The format of the FINDSJRNL control line is:

```
CALL BLGTSX 'FINDSJRNL',searchword,searchtype,occurrence
```

Parameter Descriptions

1. searchword

Valid reply

The string to be searched for. Valid values for searchword vary based on the search type.

DATA 1 to 32 characters

SWORD

2 to 10 characters, with the first between X'BA' and X'BF'. The TSX READDICT control line can be used to get the s-word from the dictionary. Refer to "READDICT" on page 160 for more information on the TSX READDICT control line.

CSDATA

1 to 32 characters

Default
None

If omitted, *any* item will match the search.

Optional

2. searchtype

Valid reply
The type of search.

DATA Search for a history data field containing the indicated prefix/data or unprefixed data.

CSDATA
A case-sensitive search for a history data field containing the indicated prefix/data or unprefixed data.

SWORD
Search for a history data field with the specified s-word.

Default
DATA

Optional

3. occurrence

Valid reply
Which item to search for.

NEWEST
The most recent history item containing the searchword.

OLDER
The next older entry (following the item last found) containing the searchword.

OLDEST
The oldest history item containing the searchword.

Default
OLDER

Optional

Usage Notes and Examples

This is an example of using a FINDSJRNL control line in a TSX. The oldest occurrence of the assignee name is located in the history data of the record.

```
CALL BLGTSX 'FINDSJRNL','PERA/.','DATA','OLDEST';
```

Return and Reason Codes

After the FINDSJRNL control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 13 on page 109.

Table 13. FINDSJRNL Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0		Successful completion.
0	12	List item history data was found. TSCATLIX is updated with the index of the located item.
8	4	The requested item was not found.
8	8	Get variable data was specified in the control line but the length field for the variable data area is zero. You must set up the variable data area and its length before processing a FINDSJRNL control line that has YES in the **Get variable data** field. The TSP must either call a user exit routine that sets the variable data area length and moves data into it, or include a MOVEVAR control line. Check your TSP to make sure a USEREXIT or MOVEVAR control line appears in the processing path before the FINDSJRNL control line that resulted in the unexpected return code. If a USEREXIT control line is in the TSP, check the exit routine's code to make sure it sets the length field properly.
8	10	The sum of the lengths of the list index and the s-word to be located exceeds the length of the s-word field in the TSCA.
8	12	An internal logic error occurred. Contact your Tivoli representative.
8	16	The data being used to build the search argument is too long. The maximum length of the search argument is 32 characters. To correct the problem, update your TSP and make sure that the search argument is no longer than 32 characters.
8	24	The user data is not a valid mixed string. Check the data and make the changes required to ensure that you specify valid mixed data.
8	28	An internal logic error occurred in National Language Support. Contact your Tivoli representative.

You can use the TESTFIELD control line to test for these return and reason codes. You can also use TESTFIELD to examine the returned data. See "TESTFIELD" on page 175 for information on how to do this.

TSCA Field Usage

Tivoli Information Management for z/OS sets the following TSCA fields after a FINDSJRNL control line is run. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

TSCAFRET
: Function return code

TSCAFRES
: Function reason code

TSCAVPHL
Visible phrase length
TSCAVPH
Visible phrase
TSCARSDL
S-Word length
TSCARSD
S-Word
TSCARPDL
P-Word length
TSCARPD
P-Word
TSCASDFL
Data field length
TSCASDF
Structured data field
TSCATLIX
List index field.

In a TSP if you enter **YES** in the **Get variable data** field and you set the variable data with a user exit routine, the user exit routine must set the length of the variable data in the following TSCA field:
TSCAVDAL
Current user variable data length

If you set the variable data with the TSP MOVEVAR control line, the MOVEVAR control line sets the field length of TSCAVDAL for you.

FINDTEXT (GETTEXT)

This control line enables you to locate freeform text in the current record using a specified s-word. The text is written to a compound variable. FINDTEXT and GETTEXT provide identical function and have identical parameters. Information on GETTEXT and the GETTEXT control line can be found in “GETTEXT (FINDTEXT)” on page 123.

This control line, like the GETTEXT control line, can be called only from a TSX.

The FINDTEXT Control Line

The format of the FINDTEXT control line is:

```
CALL BLGTSX 'FINDTEXT',sword,stemname
```

Parameter Descriptions

The Parameter Descriptions for the FINDTEXT control line are identical to the Parameter Descriptions for the GETTEXT control line, described in “The GETTEXT Control Line” on page 123.

Return and Reason Codes

The Return and Reason Codes for the FINDTEXT control line are identical to the Return and Reason Codes for the GETTEXT control line, described in “Return and Reason Codes” on page 125.

FLATTEN

This control line enables you to selectively copy records to a buffer from a Tivoli Information Management for z/OS database or a user database that has the same format as a Tivoli Information Management for z/OS database. Copying does not modify the data in the original record. The database identified in your user profile is searched for the record that you want flattened. For more information about user profiles, refer to the *Tivoli Information Management for z/OS User's Guide*.

CAUTION:
If you do not use the FLATTEN control line correctly, it could damage your existing databases. You may want to take precautions against its unauthorized use. For information on the security measures you can use to protect against its misuse, refer to the discussion in the *Tivoli Information Management for z/OS Planning and Installation Guide and Reference* **regarding data integrity and security using TSPs and TSXs .**

Refer to "The FLATTEN TSX Control Line" on page 114 for information about using the FLATTEN control line in a TSX.

After Tivoli Information Management for z/OS copies the flattened record to the buffer, you must call your exit routine to take that record and write it to auxiliary storage. Tivoli Information Management for z/OS cannot access the record that is in auxiliary storage. If you need to access the flattened record, you must restore it to a Tivoli Information Management for z/OS database using the UNFLATTEN control line. See "UNFLATTEN" on page 188 for more information on using the UNFLATTEN control line.

After a record is stored outside the original Tivoli Information Management for z/OS database, you can delete the original record to give you more space in your database.

Before using this control line, read about the security measures you can use to protect against the misuse of this control line. Refer to the *Tivoli Information Management for z/OS Planning and Installation Guide and Reference* for a discussion of data integrity and security using TSPs.

This is an example of using a FLATTEN control line in a TSP that copies from a database all problem records that have been closed before a specified date.

```
USEREXIT   OPENSEQF
ADDDATA    6,1,SE + STAC/CLOSED         (Entered in literal field)
ADDDATA     DATR/2001/01/01 -2001/10/01 (Entered in literal field - if there
                                         was no space before DATR, the search
                                         would result in STAC/CLOSEDDATR...)
PROCESS    ERROR
LABEL      ERROR
TESTFLOW   ALLDONE                   (If no records found, message BLG19214,
                                     take TRUE branch to ALLDONE)
TESTFIELD  JUSTONE                   (If TSCATPLC =1, take TRUE branch to JUSTONE)

ADDDATA    LINECMD SS,DOWN LAST,LINECMD SS  (Entered in literal field)
PROCESS    ERROR
BRANCH     NEXT
LABEL      JUSTONE
ADDDATA    LINECMD S                (Entered in literal field)
PROCESS    ERROR
LABEL      NEXT
TESTFIELD  ERROR                    (If TSCAFRET ¬=0, take TRUE branch to ERROR)
FLATTEN
USEREXIT   WRITFLAT
```

```
TESTFIELD  ERROR                  (If TSCAFRET ¬=0, take TRUE branch to ERROR)
ADDDATA    ;END                   (Entered in literal field)
PROCESS    ERROR
TESTFLOW   NEXT                   (If panel is BLG0S010, take TRUE branch to NEXT)
LABEL      ALLDONE
USEREXIT   CLOSSEQF
RETURN
LABEL      ERROR
PRINT                             (Print messages, panel, and TSCA)
BRANCH     ALLDONE
```

Creating a FLATTEN Control Line

You use the following FLATTEN Specification panel to create a FLATTEN control line.

```
BLM8CU9Z                FLATTEN SPECIFICATION                PANEL: ________

Enter 'FLATTEN' control data; cursor placement or input line entry allowed.



 1. Use id of current record...... YES
 2. Use id of last record filed... NO_
 3. Get variable data............. NO_
 4. Literal data.................. ____________________________________


     When you finish, type END to save or CANCEL to discard any changes.









===>
```

General Rules

Because of the USEREXIT considerations listed in “USEREXIT” on page 193, use assembler language to write exit routines to perform the file operations associated with the FLATTEN control line.

Field Descriptions

1. Use ID of current record

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that the ID of the record to be taken from the database and flattened is that of the current record. If you enter **NO** or make no reply, some other record ID (specified in one of the other fields on this panel) is used.

Default

YES

Restrictions

To have a current record ID, you must access a record before processing this control line.

If you enter **YES** in this field, you must enter **NO** or make no reply to the other fields on this panel.

2. Use ID of last record filed

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that the ID of the record to be taken from the database and flattened is that of the last record filed. If you enter **NO** or make no reply, some other record ID (specified in one of the other fields on this panel) is used.

Default

NO

Restrictions

To have a record in this field, you must file a record before processing this control line. If you enter **YES** in this field, you must enter **NO** or make no reply to the other fields on this panel.

3. Get variable data

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want the ID of the record to be flattened taken from the variable data area in the TSCA. If you enter **NO** or make no reply, some other record ID (specified in one of the other fields on this panel) is used.

Default

NO

Restrictions

If the FLATTEN control line has **YES** in this field, a user exit routine must move data into the variable data area and set the variable data length or the MOVEVAR control line must move data into the variable data area before processing the FLATTEN control line. If you enter **YES** in this field, you must enter **NO** or make no reply in the other fields on this panel.

If the FLATTEN control line has **YES** in this field, a record ID must be moved into the first 8 bytes of the variable data area and set the variable data length before processing a FLATTEN control line. The TSCA contains fields for a pointer to the variable data area (TSCAVDAP) and for the length of that data (TSCAVDAL). The pointer field contains the address of a variable data area that is allocated when the TSP environment is initialized. Your exit routine must not modify this pointer.

4. Literal data

Valid reply

A string of 1 to 32 characters or no reply.

This field specifies the ID of the record to be taken from the database and flattened. Any other reply indicates that some other record ID (specified in one of the other fields on this panel) is used.

Default

No reply.

Restrictions

If you enter data in this field, you must enter **NO** or make no reply in the other fields on this panel. You must enter the record ID in the first 8 positions of the field.

Usage Notes

The FLATTEN control line operates only on the single record you identify. If you are copying a set of parent and child records or related records (such as change and related activity records), a separate FLATTEN control line, followed by a user exit routine, must be issued for each record.

Each record must have a unique record ID (RNID/). Using the FLATTEN control line on a record ID that is not unique could cause an ABEND code to be returned.

The FLATTEN control line always reads the record to be flattened from the database, not from current storage. If you want to change the record before it is flattened, your TSP must change the record and file it before the FLATTEN control line is run. If not, the unchanged record is copied.

What the Control Line Does

When the TSP is run, the record ID is obtained from the area you indicated on the FLATTEN Specification panel. The record is read from the Tivoli Information Management for z/OS database. Tivoli Information Management for z/OS obtains a buffer to hold the flattened form of the record, and the flatten buffer pointer (TSCAFBP) and length (TSCAFBL) fields in the TSCA are set. After these fields are set, subsequent USEREXIT control lines in the TSP can call exit routines that access this information.

To access your information in this buffer, you must determine the length of the data to be processed. The flatten buffer is a variable length data field. You must extract from this buffer the value in the full word located at offset 24 (X'18'). Add 14 (X'E') to that value to get the length of all variable length data from the beginning of the buffer to the end of the data. This is the length of the data that must be processed by a user exit that receives the flattened buffer.

Note: The length of the data will exceed 32K (32 767) bytes in some cases. If it does, your FLATTEN exit routine must segment the data into multiple records in the flattened file. The corresponding UNFLATTEN exit routine must reassemble the segmented flattened records into the unflatten buffer.

The FLATTEN TSX Control Line

The format of the FLATTEN control line is:

```
CALL BLGTSX 'FLATTEN',rnid,stemname,linelen,options
```

Parameter Descriptions

1. rnid

Valid Reply

The RNID of the record to be flattened.

Default

None

Required

2. stemname

Valid reply

The stem of a REXX compound variable that the flattened record will be written to. The number of lines written to the compound variable is contained in *stemname.0*. EXECIO can be used to write the data stored in the compound variable to a file.

Default

BLG_FLATTEN.

Optional

3. linelen

Valid reply

Indicates the maximum number of bytes of the flattened record to be stored in a single variable. It should be the record length of the file to which you will be writing the data and cannot exceed 32752.

Default

255

Optional

4. options

Valid reply

Options associated with the record to be flattened. You can specify both NOHISTORY and NOTEXT, in either order, separated by a comma.

NOHISTORY

Indicates that the history data for this record is not copied when the flattened record is written to the buffer. The default (no value) is to copy the history data with the record.

NOTEXT

Indicates that the freeform text data for this record is not copied when the flattened record is written to the buffer. The default (no value) is to copy the freeform text with the record.

no value specified

History data is copied with the record and freeform text is copied with the record.

Default

No value specified

Optional

Usage Notes and Examples

CAUTION:

Do not modify the contents of the compound variable that the flattened record is stored in if you intend to use the UNFLATTEN control line to unflatten the record.

This is an example of using a FLATTEN control line in a TSX. Also refer to the BLGFLAT data set member of the SBLMTSX data set.

```
rnid='00003168'
member='R0003168'
```

```
/* Flatten record using the default stem with 133-byte lines       */
CALL BLGTSX 'FLATTEN',rnid,,133

if (tscafret=0) then do       /* If FLATTEN succeeded              */
  /* Allocate PDS member and write the flattened record into it    */
  'ALLOC FI(FLATPDS) DA('BLM.FLATPDS('member')'') SHR'
  'EXECIO' BLG_FLATTEN.0 'DISKW FLATPDS (FINIS STEM BLG_FLATTEN.'
  if rc=0 then do             /* If write successful, set message  */
    message='Record' rnid 'flattened successfully.'
    CALL BLGTSX 'MESSAGE',,message /* Issue message                */
    end                       /* End PDS write successful          */
  'FREE FI(FLATPDS)'          /* Free PDS member                   */
  end                         /* End flatten successful            */
```

Return and Reason Codes

After a FLATTEN control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 14.

Table 14. FLATTEN Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion.
8	4	The record cannot be read from the indicated database because of an I/O error. Contact your system programmer.
8	8	The record ID cannot be found in the indicated database. Tivoli Information Management for z/OS attempts to read the specified record from the default database. If you did not specify a default database in your profile, Tivoli Information Management for z/OS attempts to read the record from the Tivoli Information Management for z/OS database. Make sure your profile is set to select the correct database for the record ID you want to flatten and that you use the correct session member.
8	12	The length of the record ID is 0. If you want the current record flattened, your TSP must access the record before running this control line. If you want to flatten the last record filed, your TSP must file the record before running this control line. If you entered YES in the **Get variable data** field, check to see if your TSP uses the variable data area length field properly and that it moved data into the variable data area. You must set up the variable data area and its length before running this control line. You can call a user exit routine that sets the variable data area length and moves data into it or use a MOVEVAR control line.
8	16	The storage available is not sufficient to flatten the record. Increase the region size and try again.

Table 14. FLATTEN Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	20	The record ID supplied in the variable data area or the **Literal data** field was longer than 8 characters. If you created the control line using literal data, check the **Literal data** field on the FLATTEN Specification panel to make sure it contains 8 or fewer characters. If you entered YES in the **Get variable data** field, check your exit routine to see if the current user variable data length field (TSCAVDAL) is set properly.
8	24	Your TSP attempted to flatten a record in Information/MVS format. MVS™ records are read-only records and cannot be unflattened. The FLATTEN control line does not accept MVS records to be flattened.
8	28	The record that you want to flatten is being used and cannot be flattened now. Try again later when the record is not in use.
8	32	An internal logic error occurred in National Language Support. Contact your Tivoli representative.
8	36	The user data is not a valid mixed string. Check the data and make the changes required to ensure you specify valid mixed data.

You can use the TESTFIELD control line to test for these return and reason codes. See "TESTFIELD" on page 175 for information on how to do this.

TSCA Field Usage

Tivoli Information Management for z/OS sets the following TSCA fields after a FLATTEN control line is run. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289

.

TSCAFRET
: Function return code

TSCAFRES
: Function reason code

TSCAFBL
: Flatten buffer length

TSCAFBP
: Pointer to flattened record

If you enter **YES** in the **Get variable data** field and you set the variable data with a user exit routine, the user exit routine must set the length of the variable data in the following TSCA field:

TSCAVDAL
: Current user variable data length.

If you set the variable data with the MOVEVAR control line, the MOVEVAR control line sets field TSCAVDAL for you.

GETAPIDATA

This control line can be used in a TSX to receive parameters that were passed as input PDBs via HLAPI transaction HL14 (Start User TSP or TSX). The HL14 transaction starts a user TSP or TSX and passes a parameter to it.

If the HL14 starts a TSX, the invoked TSX can use the GETAPIDATA control line to access the data specified in any of the three reserved input PDBs that the HL14 accepts. The reserved input PDBs that an HL14 accepts are TSP_NAME, USER_PARAMETER, and USER_PARAMETER_DATA. These PDBs will not be passed to the TSX. For example, USER_PARAMETER_DATA can be used to pass a parameter to a TSX. You can also use GETAPIDATA to pass other data to the TSX; these input PDBs can be named whatever you choose. If the PDB name is found, the PDB data is returned in a REXX stem variable. The *Tivoli Information Management for z/OS Application Program Interface Guide* contains additional information about the HLAPI and PDBs.

Each call to GETAPIDATA causes the input PDB chain to be searched for the specified PDB name. If the PDB name is found, the PDB data is returned in a REXX stem variable. If the PDB name appears in the input chain multiple times (for example, USE_AS_IS_ARGUMENT), the data from all the found PDBs will be returned in the same stem variable. Trailing blanks will be removed when data is returned in the stem variable. Thus, each element in the stem variable can have a different length.

This control line can be called only from a TSX and is valid only in an API environment. LLAPI applications using the T111 transaction will not be able to pass input PDBs to a TSX.

The GETAPIDATA Control Line

The format of the GETAPIDATA control line is:

```
CALL BLGTSX 'GETAPIDATA',pdbname,stemname
```

Parameter Descriptions

1. pdbname

Valid reply

The name of the PDB to search for in the input PDB chain. Maximum length is 32 characters.

Default

None

Required

2. stemname

Valid reply

The name of a REXX compound variable that will contain the data read from PDBDATA. By convention, the variable name should end with a period. The length is limited to 58 characters, including the period.

Default

BLG_GETAPIDATA.

Optional

Usage Notes and Examples

If the input PDB contains a single string (PIXDDATW = 0 and PIXDDATL > 0), then PIXDDATL must be less than or equal to 32767. If the string is longer than 32767, the data will be truncated and both TSCAFRET and TSCAFRES will be set to 4.

If the input PDB contains a series of text lines (PIXDDATW > 0 and PIXDDATL > 0), then PIXDDATW must be less than or equal to 255. However, PIXDDATL can be greater than 32767 in this case. If any text line is longer than 255, the data will be truncated and both TSCAFRET and TSCAFRES will be set to 4.

This is an example of using a GETAPIDATA control line in a TSX using an s-word as a pdbname.

```
CALL BLGTSX 'GETAPIDATA','S0B59','MYARRAY.'
```

This is an example of using a GETAPIDATA control line in a TSX using a user-named pdbname.

```
CALL BLGTSX 'GETAPIDATA','INPUTFIELD','MYARRAY.'
```

Return and Reason Codes

After the GETAPIDATA control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 15.

Table 15. GETAPIDATA Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Processing successful. Data was copied from the input PDB into the REXX stem variable.
4	4	Processing successful. Data was copied from the input PDB into the REXX stem variable but was truncated.
8	4	The specified PDB name was not found in the input PDB chain.
8	8	Data could not be copied to the REXX stem variable.
8	12	API not active.

GETLIST

This control line retrieves all or part of a list processor list into a REXX compound variable.

This control line can be called only from a TSX.

The GETLIST Control Line

The format of the GETLIST control line is:

```
Call BLGTSX 'GETLIST',listsword,stemname,startln,lastln
```

Parameter Descriptions

1. listsword

Valid reply

The list s-word (or root s-word) of the list to be retrieved. The list s-word includes the hexadecimal watermark character; therefore, it is recommended that you use the

TSX READDICT control line to get the root s-word that you will pass to GETLIST. Refer to "READDICT" on page 160 for more information on the READDICT control line.

Default
None

Required

2. stemname

Valid reply
The stem of a REXX compound variable that the list data will be written to. The index of the highest list index is contained in *stemname.0* whether or not you retrieve the entire list. EXECIO can be used to write the data stored in the compound variable to a file.

Default
BLG_LIST.

Optional

3. startln

Valid reply
A decimal number between 1 and 19274 representing the first item to be retrieved.

Default
An index of 1

Optional

4. lastln

Valid reply
A decimal number between 1 and 19274 representing the last item to be retrieved.

Default
None

If omitted, all items from *startln* to the last non-blank item in the list will be retrieved.

Optional

Usage Notes and Examples

CAUTION:
The list processor data that the TSX GETLIST control line retrieves must be in the internal format introduced in Version 5.1 (or in Version 4 by APARs OY47188 and OY47893). Lists which were stored prior to the internal format change will not be retrieved accurately by GETLIST. Those lists can be converted to the new internal format by updating them in Version 7.1, repeating the first line, deleting the first line, and filing the record. The data will be unchanged, but it will be stored in the correct format.

This is an example of using a GETLIST control line in a TSX.

```
sword='S1416';
CALL BLGTSX 'READDICT',SWORD;   /* Get the s-word.             */
rootsword=TSCARSD;   /* The root s-word of the LP data.        */
BLG_LIST.  = '';
BLG_LIST.0 =  0;
CALL BLGTSX 'GETLIST',ROOTSWORD;
/* The LP data is stored in REXX compound variable BLG_LIST.   */
/* The index of the highest list entry is stored in BLG_LIST.0.*/
```

This is another example of using a GETLIST control line in a TSX. The first two list entries for the root s-word identified by s-word index X'1416' will be read into compound variable MY_LIST.

```
sword='S1416';
CALL BLGTSX 'READDICT',SWORD;   /* Get the s-word.             */
rootsword=TSCARSD;   /* The root s-word of the LP data.        */
MY_LIST.  = '';
MY_LIST.0 =  0;
CALL BLGTSX 'GETLIST',ROOTSWORD,'MY_LIST.',1,2;
/* The LP data is stored in REXX compound variable MY_LIST .   */
/* The index of the highest list entry is stored in MY_LIST.0. */
```

The *stemname.0* for the compound variable will be set with the highest index of any item in the list, or to 0 if there are no items in the list.

For a request to retrieve the entire list, the stem is initialized to null before processing; for partial list retrieval requests, it is the responsibility of the TSX writer to initialize the stem if desired. The GETLIST control line only sets variables for non-blank list items. Therefore, for partial list retrieval requests, variables for blank list items will retain the value they had prior to calling GETLIST which could be ″uninitialized″ or even a value set by a previous GETLIST, so it is important for your TSX to ensure that the stem is initialized prior to the first call to GETLIST for a given list.

Note: GETLIST will retrieve items from the modified list if the current panel is a list processor table panel which contains the list being retrieved. This means that calling GETLIST if the current panel is an assisted-entry panel or help panel will retrieve the “saved” list, even if an update session for the list is active.

This is an example of how to initialize the stem prior to the first call to GETLIST.

```
BLG_LIST.='';
```

Return and Reason Codes

After the GETLIST control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened.

Using GETLIST is very much like using an enhanced FINDSDATA (see “FINDSDATA” on page 87). The return and reason codes for GETLIST are the same as the return and reason codes for FINDSDATA. Refer to Table 16 for exceptions and modifications.

Table 16. GETLIST Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Processing successful - one or more non-blank items found in specified range of list.

Table 16. GETLIST Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	4	Processing successful - list exists, but there are no non-blank items in the specified range. Stemname.0 is set.
8	4	No list items exist for the specified root s-word. Stemname.0 is set to 0.

GETRDATA

This Remote Data Resource TSX control line is described in "GETRDATA" on page 242.

GETSCREEN

This control line can be used in a TSX to retrieve data from any Tivoli Information Management for z/OS screen that can be displayed. For example, you might use GETSCREEN to capture screens from a Search Results List and return columns of data to a HLAPI. A call to GETSCREEN will return the contents of a single screen.

This control line can be called only from a TSX. An ISPF or TSO environment is not required for using GETSCREEN.

The GETSCREEN Control Line

The format of the GETSCREEN control line is:

```
CALL BLGTSX 'GETSCREEN',stemname
```

Parameter Descriptions

1. stemname

Valid reply

The name of a REXX compound variable (including a separator character, such as a period). By convention, the variable name should end with a period. The length is limited to 58 characters, including the period. On return from GETSCREEN, each stem_name.N variable returned will contain the data from the Nth line of the screen, where N is the value 1 to the value returned in stem_name.0.

Default

BLG_SCREEN.

Optional

Usage Notes and Examples

This is an example of using a GETSCREEN control line with no parameters in a TSX. The default stem variable BLG_SCREEN. contains the current screen. BLG_SCREEN.0 is the number of lines of the currently displayed screen and each BLG_SCREEN.n variable corresponds to the physical line number of the Tivoli Information Management for z/OS screen, including the command line when the command line is located at the top. Screen attributes are translated to blanks (X'40').

```
CALL BLGTSX 'GETSCREEN'
```

This is an example of using a GETSCREEN control line and specifying MYSTEM. as a stem variable. If you ran this example while viewing the PMF panel list BLM1TPDS, the TSX would show the list of panel data sets. The stem variable MYSTEM. contains the

current screen. MYSTEM.0 is the number of lines of the currently displayed screen, including the command line when the command line is located at the top and each MYSTEM.n variable corresponds to the physical line number of the Tivoli Information Management for z/OS screen. Screen attributes are translated to blanks (X'40').

```
CALL BLGTSX 'GETSCREEN','MYSTEM.'
```

Note: The ISPF areas (the Menu bar when using the enhanced panels, the PFKEY area, and any split part of the screen for the other split screen session) are not included in the stem variable. The last two lines of the screen, which are reserved, are not included in the stem variable. This is identical to the way that the TSX/TSP PRINT line works.

Return and Reason Codes

After the GETSCREEN control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields. For the GETSCREEN command line, these are always returned as 0. If GETSCREEN fails, the REXX syntax exception will be set.

GETTEXT (FINDTEXT)

This control line enables you to locate freeform text in the current record using a specified s-word. The text is written to a compound variable. GETTEXT and FINDTEXT provide identical function and have identical parameters.

This control line can be called only from a TSX.

The GETTEXT Control Line

The format of the GETTEXT control line is:

```
CALL BLGTSX 'GETTEXT',sword,stemname,startln,lastln
```

Parameter Descriptions

1. sword

Valid reply

2 to 10 characters, with the first between X'BA' and X'BF'. The TSX READDICT control line can be used to get the s-word from the dictionary. Refer to “READDICT” on page 160 for more information on the TSX READDICT control line.

Default

None

Required

2. stemname

Valid reply

The stem of a REXX compound variable that the freeform text will be written to. The number of lines written to the compound variable is contained in *stemname.0*. EXECIO can be used to write the data stored in the compound variable to a file.

Default

BLG_TEXT.

Optional

3. startln

Valid reply
The value specified indicates the line number of the first line of text to retrieve. Valid values are 1 to 999999.

Default
1

Optional

4. lastln

Valid reply
The value specified indicates the line number of the last line of text to retrieve. Valid values are 1 to 999999.

Default
999999

Optional

Usage Notes and Examples

This is an example of using a GETTEXT control line in a TSX.

```
sword='S0E01';
CALL BLGTSX 'READDICT',sword;   /* Get the s-word.            */
textsword=TSCARSD;   /* The s-word of the text data.          */
BLG_TEXT.  = '';
BLG_TEXT.0 =  0;
CALL BLGTSX 'GETTEXT',textsword;
/* The text is stored in REXX compound variable BLG_TEXT.     */
/* The number of elements in the array is stored in BLG_TEXT.0.*/
```

Note: For a request to retrieve all text lines, the control line initializes the stem to null before processing; for partial text retrieval requests, it is the responsibility of the TSX writer to initialize the stem. The GETTEXT control line only sets variables for non-blank text lines. Therefore, for partial text retrieval requests, variables for blank text lines will retain the value they had prior to calling GETTEXT. This variable could be uninitialized, or could be a value set by a previous GETTEXT, so it is important for your TSX to ensure that the stem is initialized before the first call to GETTEXT for a given type of text.

The *s-word* parameter specified is for freeform text. An optional start line parameter *first* can be specified after the output stem name and an optional end line parameter *last* can be specified after the start line number to limit the lines of text retrieved.

When the GETTEXT control line is called from the Tivoli Information Management for z/OS freeform text editor screen for the specified text type, the current text from that edit session will be retrieved even if it has not yet been saved.

Note: GETTEXT will retrieve modified text only if the current panel is a freeform text table panel which contains the text being retrieved. If the current panel is anything else, such as an assisted-entry panel or a help panel, calling GETTEXT will retrieve the saved text, even if an edit session for the text is active.

This shows another example of using a GETTEXT control line in a TSX.

```
sword='S0E01';
CALL BLGTSX 'READDICT',SWORD;   /* Get the s-word.            */
textsword=TSCARSD;   /* The s-word of the text data.          */
```

```
MY_TEXT.  = '';
MY_TEXT.0 =  0;
CALL BLGTSX 'GETTEXT',TEXTSWORD,'MY_TEXT.',1,5;
/* The text is stored in REXX compound variable MY_TEXT.       */
/* The number of elements in the array is stored in MY_TEXT.0. */
/* This will retrieve the first five lines of text             */
/* The first line of text is stored in MY_TEXT.1         */
/* The second line of text is stored in MY_TEXT.2, etc.  */
```

Return and Reason Codes

After the GETTEXT control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 17.

Table 17. GETTEXT Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Processing is successful. Freeform text is written to the compound variable.
0	4	Processing is successful. There is text for the specified s-word, but none exists in the specified range. No text lines are retrieved, but *stem.0* is set to the total number of lines of text for the s-word.
8	4	The freeform text was not found. The compound variable is unchanged.

ISPEXEC

This control line enables you to call ISPF dialog management services from within a TSP. You can use ISPEXEC to run CLISTs or to call ISPF applications that could otherwise not be run in a TSP environment.

The functions provided by the TSP ISPEXEC control line are available in a TSX using the standard program flow controls available in the REXX programming language. Please note, however, that you cannot perform ISPF variable services from within a TSX. In order to perform ISPF variable services from a TSX, you must call another EXEC to perform the services. An example call:

```
/* A REXX TSX*/
Say '---- Call ISPEXEC SELECT CMD()'
address ISPEXEC 'SELECT CMD(EX 'DEWRIG.TSX(ISPFSERV)'' ''A B C'')'
say 'rc='rc
exit
```

You can also use the USEREXIT control line and user exits BLGSPFGT and BLGSPFPT. Refer to “User Exits” on page 263 for more information on the BLGSPFGT and BLGSPFPT user exits.

For more syntax information on ISPF services, refer to the *ISPF Dialog Management Services* manual.

If you want to display the ISPF primary options menu, specify:

```
SELECT PANEL(ISR@PRIM)
```

The ISPEXEC control line generates the call to ISPF module ISPEXEC and passes the data to ISPF without verifying that the ISPF function specified is valid.

Creating an ISPEXEC Control Line

Use the following ISPEXEC Specification panel to create an ISPEXEC control line:

```
BLM8CU9I                ISPEXEC SPECIFICATION                PANEL: ________

Enter 'ISPEXEC' control data; cursor placement or input line entry allowed.




 1. Get variable data.............. NO_
 2. Literal data................... SELECT PANEL (ISR@PRIM)_________


     When you finish, type END to save or CANCEL to discard any changes.









===>
```

General Rules

You must enter **YES** in the **Get variable data** field or enter information in the **Literal data** field.

Field Descriptions

1. Get variable data

Valid reply

YES or **NO**

This field indicates whether you want this control line to use variable data extracted from the TSCA.

Default

NO

Restrictions

The TSCA contains fields for a pointer to the variable data area (TSCAVDAP) and the length of that data (TSCAVDAL). These fields must be set before the processing of a control line with **Get variable data** set to **YES**. If the ISPEXEC control line has **YES** in this field, a user exit routine must move data into the variable data area and set the variable data length, or a MOVEVAR control line must move data into the variable data area before processing the ISPEXEC control line.

You cannot enter **YES** in this field if you enter data into the **Literal data** field.

For examples of TSPs using the ISPEXEC control line, use PMF to look at TSPs BTNTCN01 and BTNTPN01 in the base panel data set.

2. Literal data

Valid reply

A string of 1 to 32 characters or no reply.

This field specifies the static data to be passed to the ISPF dialog manager. You can use any valid call invocation syntax from ISPF dialog management services to specify data in this field.

Default

No reply.

Restrictions

You cannot specify data in this field if you enter **YES** in the **Get variable data** field.

Usage Notes

When using a CLIST, specify any data, whether from the variable data area or the literal data area, with valid syntax for the ISPF dialog management service, leaving off the leading ISPEXEC. This data is converted to a buffer and passed with the calculated length to ISPEXEC.

What the Control Line Does

When the TSP is run, the data to be passed to ISPF is extracted from the control line or from the variable data area. The length of the data is calculated, and a call to ISPF's ISPRXEC dialog management services is made with the buffer and the buffer length parameters specified.

Note: The ISPEXEC control line cannot be used in TSPs that are running under one of the application program interfaces because the ISPF environment has not been established.

Return and Reason Codes

After the ISPEXEC control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 18.

Table 18. ISPEXEC Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0		Successful completion.

Table 18. ISPEXEC Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	8	Get variable data is specified in the control line, but the length for the variable data area (TSCAVDAL) is zero. Your TSP must set up the variable data area and its length before you can run an ISPEXEC control line that has YES in the **Get variable data** field. You can set the variable data area and its length by calling a user exit routine that sets the variable data area length and moves data into the variable data area, or by using a MOVEVAR control line. Check your TSP to make sure a USEREXIT or MOVEVAR control line appears in the processing path before the ISPEXEC control line that caused the unexpected return code. If a USEREXIT control line is in the TSP, check the exit routine's code to make sure that it sets the length field properly. If you set the variable data area using the MOVEVAR control line, add the TESTFIELD control line after the MOVEVAR control line to verify that the return code is 0 before you continue.
8	12	An internal logic error occurred in National Language Support. Contact your Tivoli representative.
8	16	The user data is not a valid mixed string. Check the data and make the changes required to ensure that you specify valid mixed data.
12		The ISPEXEC function failed. The reason code is set to the return code from the ISPF dialog management service that was called. Refer to the description of the service that you want to use in the *ISPF Dialog Management Services* manual.

You can use the TESTFIELD control line to test for these return and reason codes. See "TESTFIELD" on page 175 for information on how to do this.

TSCA Field Usage

The following TSCA fields are set by Tivoli Information Management for z/OS when an ISPEXEC control line is processed. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

TSCAFRET
: Function return code

TSCAFRES
: Function reason code.

If you enter **YES** in the **Get variable data** field and you set the variable data with a user exit routine, the user exit routine must set the length of the variable data in the following TSCA field:

TSCAVDAL
: Current user variable data length.

If you set the variable data with the MOVEVAR control line, the MOVEVAR control line sets field TSCAVDAL for you.

LABEL

This control line enables you to identify the target of another control line such as BRANCH, TESTFIELD, or PROCESS. It also enables you to make notes or comments about the TSP.

The function provided by the TSP LABEL control line is available in a TSX using the standard program flow controls available in the REXX programming language.

You can use a LABEL control line to separate control lines of the TSP into subsections, or to provide unique checkpoint identifiers to use when you review output from the TRACE function.

This is an example of the LABEL control line used as a checkpoint identifier:

```
LABEL      BEGIN          THIS IS THE BEGINNING
TRACE
LABEL      SETUP
⋮

LABEL      UPDATE         UPDATE THE CURRENT RECORD
⋮

LABEL      CLEANUP
⋮
```

Whenever you see label names in your trace output, you know what set of control lines was being processed.

Creating a LABEL Control Line

Use the following LABEL Specification panel to create a LABEL control line:

```
BLM8CU9G                  LABEL  SPECIFICATION              PANEL: ________

Enter 'LABEL' control data; cursor placement or input line entry allowed.



                  1. Label name... BEGIN
                  2. Literal data. THIS IS THE BEGINNING




     When you finish, type END to save or CANCEL to discard any changes.









===>
```

Field Descriptions

1. Label name

Valid reply

A mixed string of 1 to 8 characters, beginning with an SBCS alphanumeric, national character, or an SO character.

This field identifies the name of this control line. A LABEL control line is usually the target of another control line, such as PROCESS or BRANCH.

Default

None.

Restrictions

The first character cannot be numeric. The label name must be unique for this TSP.

2. Literal data

This field is not used to process TSPs. It only exists for you to make notes or comments about the TSP.

Usage Notes

You can use the LABEL control line as a target marker for other control lines that perform branching functions if you specify the **Label name** field. When a TSP is filed, the line index of a referenced label is reflected in the control line that references the label.

What the Control Line Does

When the TSP is run, the LABEL line is ignored. The one exception to this is its entry in the TRACE report.

Return and Reason Codes

Running a LABEL control line does not change the TSCA return and reason codes.

LINK

This control line allows you to transfer control to another TSP by naming the TSP to be called when the control line is run.

You can also use the LINK control line to call a terminal simulator EXEC. You may wish to store parameters in the variable data area (VDA) by using the MOVEVAR control line in a TSP (described in “MOVEVAR” on page 143) prior to invoking the LINK control line. The contents of the VDA are passed as a single argument to the TSX when it is started. Refer to “The LINK TSX Control Line” on page 132 for additional information on using the LINK control line within a TSX.

This is an example of using the LINK control line to call a second TSP to include a set of control lines that is repeated several times in the first TSP. Each time the first TSP needs to run the repetitive control lines, it links to the second TSP.

The calling TSP:

```
LABEL    TSP00001
⋮

LINK     TSP00002
⋮
```

```
LINK     TSP00002
:

LINK     TSP00002
:

LABEL    ENDTSP1
```

The linked-to TSP:

```
LABEL    TSP00002
:

RETURN
```

Creating a LINK Control Line

Refer to Figure 4 for an example of the Specification panel used to create a TSP LINK control line:

```
BLM8CU9D                  LINK  SPECIFICATION                PANEL: TSP00001

Enter 'LINK' control data; cursor placement or input line entry allowed.


               1. Terminal simulator name....<R> TSP00002




     When you finish, type END to save or CANCEL to discard any changes.




===>
```

Figure 4. LINK Specification panel

Field Descriptions

1. Terminal simulator name

Valid reply

An 8-character TSP name or a 1- to 8-character TSX or alias name. The name must be an SBCS alphanumeric string beginning with an alphabetic character. This field is required.

This entry specifies the TSP or TSX you want to call.

Default

None.

Restrictions

The TSP or TSX must exist when the control line is run. Make sure that the BLGTSX DD is allocated so that your TSXs are accessible. Also make sure that you use PMF to copy your TSPs to the appropriate panel data set.

Usage Notes

The LINK control line sets the return and reason codes. Do not link to a TSP or TSX that analyzes return and reason codes from other functions, because these codes are always zero when the linked-to TSP or TSX receives control. Other data in the TSCA, such as s-word and p-word data, is not changed unless the linked-to TSP or TSX changes it.

If you are using TRACE in your TSPs or TSXs, specify **YES** in the Trace LINK function field on the TRACE Specification panel to ensure that tracing continues in the linked-to TSPs or TSXs.

Note: It is the responsibility of the TSX REXX code to test the value stored in the BLGTRACE variable and to issue the desired REXX TRACE statement. Refer to "TRACE" on page 185 for additional information on how to use the BLGTRACE variable to turn tracing on and off in a TSX.

A TSP can communicate with a linked-to TSP using the SETFIELD control line. The calling TSP can examine the data entered in the TSCA by the linked-to TSP. The calling TSP can also pass data to the linked-to TSP using SETFIELD. For instance, you might have one TSP that handles an error routine and is called by several TSPs. When a TSP links to this error TSP, it must identify itself through a SETFIELD control line. The error TSP can then handle the error as specified by the calling TSP, such as issuing a particular message. When returning to a TSP from a linked-to TSP, the TSCATLIX is reset to the value it had in the original TSP, unless that value is zero. If TSCATLIX was zero in the original TSP, the TSCATLIX retains the value set by the linked-to TSP.

For examples, use PMF to look at TSPs BTNTA121 and BTNTCA21 in the base panel data set.

What the Control Line Does

When the TSP or TSX is run, the following occurs:

If all steps are successful, processing continues at the first line of the linked-to TSP or TSX. When a RETURN control line is found in the linked-to TSP or after the last line of that TSP is run, or an EXIT is found in the REXX code of the linked-to TSX or after the last line of the TSX is run, processing resumes at the control line following LINK in the calling TSP or at the EXEC statement following LINK in the calling TSX.

The LINK TSX Control Line

The format of the LINK control line is:

```
CALL BLGTSX 'LINK',name,parm1
```

Parameter Descriptions

1. name

Valid reply

The name of the TSP or TSX to be linked to. This may also be an alias name.

Default

None

Required

2. parm1

Valid reply

An argument that is passed to the ″called″ TSX which that TSX can parse as needed. The data for parm1 is put into the TSCAVDA where a TSP can find it.

Default

None

Optional

Usage Notes and Examples

This is an example of using a LINK control line in a TSX. The TSX MYTSX is ″called″ from the current TSX.

```
CALL BLGTSX 'LINK','MYTSX';
```

Return and Reason Codes

After the LINK control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 19.

Table 19. LINK Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0		Successful completion.
8	4	The name you entered in the **Starting panel name** field of the Common Update panel (BLM8CU97) does not match the current Tivoli Information Management for z/OS panel. When you specify this field, Tivoli Information Management for z/OS assumes that proper processing of the TSP is based on that panel being the current panel when the TSP is started. When the starting panel and current panel do not match, the TSP cannot run. Make sure that the starting panel name specified on the Common Update panel is the name of the current Tivoli Information Management for z/OS panel that you require to be current when this TSP is started. If it is, trace the steps leading up to the TSP call to determine why the dialog was not on the expected panel when the TSP was started.
8	8	The TSP name was not found in any read panel data sets. Check to see if you specified the name correctly. If the name is correct, maybe the TSP does not exist in your read panel data sets. When you create the LINK control line, Tivoli Information Management for z/OS does not check for the existence of the TSP. As with all other panels, a TSP is initially filed in the write panel data set and must be copied to the read panel data set before running. Also check that the session member includes the name of the proper read panel data set.

Table 19. LINK Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	12	The TSP name entered on the LINK Specification panel was not loaded because of an I/O error. Retry the TSP. If another I/O failure occurs and your TSP appears to be correct, contact your system programmer.
8	16	Not enough storage existed to obtain one of the necessary buffers or control blocks. Use successive BACK or CANCEL commands to end some functions. If you are into several levels of suspension, use successive RESUME commands to end some functions. If you are in split screen, return to a single screen. If you still do not have available storage, contact your system programmer to see if your region size should be increased.
8	20	The name entered on the LINK Specification panel is a valid panel but not a TSP. Correct the name using panel update. If you are not sure of the name, request a panel list of your read panel data sets. Refer to the *Tivoli Information Management for z/OS Panel Modification Facility Guide* for information on listing panels.

You can use the TESTFIELD control line to test for these return and reason codes. See "TESTFIELD" on page 175 for information on how to do this.

TSCA Field Usage

Tivoli Information Management for z/OS sets the following TSCA fields after a LINK control line is run. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

TSCAFRET
: Function return code

TSCAFRES
: Function reason code.

MESSAGE

This control line enables you to include your own single-line message or an existing Tivoli Information Management for z/OS message in the TSP process.

Refer to "The MESSAGE TSX Control Line" on page 139 for information about using the MESSAGE control line in a TSX.

The message panel name is not required. When it is omitted, message text is taken from the variable data area or the literal data area. You can have a message added to the current messages or saved for display after the TSP session completes. For information on creating message panels, see the *Tivoli Information Management for z/OS Panel Modification Facility Guide*. See "Message Handling during TSP and TSX Processing" on page 261 for a discussion of how messages are processed in a TSP environment.

This is an example using the MESSAGE control line to indicate the success or failure of an exit routine:

```
⋮
USEREXIT   READRECS              (Readrecs sets TSCAFRET to 0 if successful;
                                  to nonzero if failure)
TESTFIELD  SUCCESS               (If the return code is zero, go to label SUCCESS)
MESSAGE    FAILPANL              (Otherwise, issue a failure message panel)
RETURN
LABEL      SUCCESS               (Exit routine completes processing)
MESSAGE    USR9E001              (Issue USR9E001 message panel)
⋮
```

Creating a MESSAGE Control Line

Use the following MESSAGE Specification panel to create a MESSAGE control line:

```
BLM8CU9T                MESSAGE SPECIFICATION                PANEL: ________

Enter 'MESSAGE' control data; cursor placement or input line entry allowed.



 1. Message panel name..... FAILPANL
 2. Save generated message........ YES
 3. Insert data type.............. CHAR
 4. Get variable data............. NO_
 5. Literal data.................. ____________________________________



     When you finish, type END to save or CANCEL to discard any changes.







===>
```

Field Descriptions

1. Message panel name

Valid reply

An 8-character message panel name. It is an SBCS alphanumeric string, beginning with an alphabetic character.

This entry specifies the message panel name for the message you want added to the current message chain or saved for display after the TSP finishes running.

Default

None.

Restrictions

The name of a message panel that contains message text must exist at the time you start the TSP. If you leave this field blank, literal data must be present, or the variable data area must contain data and you must specify **Get variable data** equal to **YES**.

2. Save generated message

Valid reply

YES, NO, or no reply.

A reply of **YES** in this field indicates that you want the specified message saved for display after the TSP finishes. Any other reply indicates that you want the message to be added to the current message chain, which is available for use while the TSP is running.

Default

YES

Restrictions

A reply of **YES** moves this message and any other messages from the current message chain to a saved message chain. The message is returned to the user when the TSP finishes. If you want to see whether a message exists on the saved message chain, use the TESTFLOW control line and look for the return code and reason code combination 4/4. This combination indicates that the message is on the saved chain.

3. Insert data type

Valid reply

CHAR, HEX, or no reply.

This field specifies that you want the data inserted into the message in either character (CHAR) or hexadecimal (HEX) format. The insert data is taken from the **Literal data** field (if used) or from the variable data area if you entered YES in the **Get variable data** field. If there is no insert data for this message, do not reply to this field.

Default

CHAR

Restrictions

You cannot enter HEX in this field if you enter data in the **Literal data** field.

4. Get variable data

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want variable data inserted into the message.

Default

NO

Restrictions

If the MESSAGE control line has **YES** in this field, a user exit routine must move data into the variable data area and set the variable data length, or the MOVEVAR control line must move data into the variable data area before processing the MESSAGE control line. If you entered **YES** in this field, you cannot enter data in the **Literal data** field.

The TSCA contains fields for a pointer to the variable data area (TSCAVDAP) and for the length of that data (TSCAVDAL). The pointer field contains the address of a variable data area that is allocated when the TSP environment is initialized. Your exit routine must not modify this pointer.

If you use the variable data area, you must specify a string of characters as long as the number of underscores you put in the message line, or 4 bytes of HEX data that

you want inserted into the message. If you add too much data to the variable data area for your message insert, the data is right truncated. For more information about inserts, see "Usage Notes".

The maximum number of characters that is possible in a message varies. You can determine the maximum number of characters enabled for a specific message by checking the message panel.

5. Literal data

Valid reply

A string of up to 32 characters, or no reply.

An entry in this field indicates that you want the entry inserted as character data. The data that is entered in this field is collected in the case entered by the user.

Default

No reply.

Restrictions

The maximum number of characters that is possible for a message varies. You can determine the maximum number of characters enabled for a specific message by checking the message panel.

If you enter data in this field, you cannot enter **YES** in the **Get variable data** field. The Insert data type must be CHAR.

Usage Notes

Unless you request it, a TSP does not display a message when it finishes. You can use the MESSAGE control line to display meaningful messages about how the TSP is running. For instance, you can end the TSP with a message that indicates a successful completion or display a message in an error routine that indicates the return code for the TSP.

If you want to insert data into a message (either from the variable data area or as literal data), build the message panel to include underscore characters in the message line. The underscores indicate where the data should be inserted.

The number of underscores must equal the maximum number of characters you anticipate for the insert data. For example, if you are using the insert field to contain a record ID, use 8 underscore characters because 8 characters is the maximum length of a record ID. If the data you insert is shorter than the number of underscores, the data is left-justified in the insert field, and trailing blanks are removed. The amount of inserted data is the smaller of the specified data and the number of underscore characters in the message panel. If you want more than 32 characters in the message, either break the message into two MOVEVAR control lines, or use a user exit routine to put the insert data into the variable data area, and then specify **YES** in the **Get variable data** field.

How long the message can be depends upon how you specify it. If you use the variable data area to enter the entire message text, the maximum length is 254 bytes. If you use a message panel for any part of the message, the maximum length is 148 bytes, including the message ID. For several examples of how to specify messages, use PMF to look at TSP BTNTAPRV in the base panel data set.

Enabling Help for Messages

Frequently, a single-line message does not give the user enough information on which to act, so you might want to supply a help panel that the user can see by issuing the HELP command. For HELP to work, you must maintain a relationship between the message panel name, message number, and help panel. Refer to the *Tivoli Information Management for z/OS Panel Modification Facility Guide* for how to create a help panel. For examples of how to enable HELP, use PMF to look at TSPs BTNTAM01 and BTNTCEA2 in the base panel data set.

Table 20. Relating Messages to Help Panels

Message panel name	xxx9ynnn	*xxx* is any 3 digits that make up the beginning of a valid panel name. *y* is an alphabetic character. *nnn* is numeric.
Message number	xxxzznnn	*xxx* and *nnn* match the corresponding parts of the message panel name. *zz* is the position in the alphabet occupied by *y*, described previously. Suppose that the message panel name is XXX*9C*111. *C* is the third letter of the alphabet, so the corresponding message number is XXX*03*111.
Help panel	xxx4ynnn	*xxx*, *y*, and *nnn* are the same as above.

What the Control Line Does

The MESSAGE control line can be used to load a panel or to convey information to another user.

- If you are using this control line to load a panel, Tivoli Information Management for z/OS loads the named panel and extracts the single-line message from the panel. If you specified an insertion, Tivoli Information Management for z/OS extracts the insert data from the control line or the variable data area, does any necessary data conversion, and merges the insert into the message. If you specify **YES** for the **Save generated message** field in the TSP (or specify the **SAVE** option in the TSX control line), the message is added to the saved message chain, which includes any messages that were pending at the time the TSP environment began.

  If you did not specify **YES** for the **Save generated message** field (or specify the **DISCARD** option in the TSX control line), the message is added to the current message chain and the TSCAMSGC field is incremented. You can then print the messages on the current message chain using the PRINT control line which is described in “PRINT” on page 148. You can also test for the messages on either the current or saved message chains using the TESTFLOW control line, described in “TESTFLOW” on page 180. When the TSP environment ends, the chain containing the saved messages merges with the current message chain, making the messages available to you.

- In addition to the functions provided with the TSP, you can also use the TSX control line (described in “The MESSAGE TSX Control Line” on page 139) to send a message to another user on the same system. In order to do this, you must specify message text, the keyword **NOTIFY**, and the name of the userid to whom you are conveying the message.

The MESSAGE TSX Control Line

The format of the MESSAGE control line is:

```
CALL BLGTSX 'MESSAGE',msgid,text,option,userid,stem,insertcount
```

Parameter Descriptions

1. msgid

Valid reply

This can be an optional Tivoli Information Management for z/OS message panel name or the numeric part of the assembler label of a message defined in CSECT BLGTMSGS.

Default

None

Optional

2. text

Valid reply

If **msgid** is specified, this is a single value to be inserted into the message. If msgid is not specified, this is the 1- to 255-character text to be used as the message. If there are no insert values or the values are specified via a REXX stem variable, this value must be omitted.

If you specify msgid, you can insert data for the message either as a single value in **text** or as one or more values in a compound variable, with the stem specified in **stem** and the count specified in **insertcount**.

Default

None

Optional

3. option

Valid reply

Options associated with the message that is to be sent. Valid options are:

SAVE|DISCARD|NOTIFY|BUILD

- **SAVE** indicates that the message should be saved for display after the TSX finishes.
- **DISCARD** indicates that the message should not be saved for display after the TSX finishes.
- **NOTIFY** indicates that the message is sent to the user specified in **userid**. If **NOTIFY** is specified, you must also specify a **userid**.
- **BUILD** causes the message to be returned to the TSX in variable TSCAVDA. The text is returned to the TSX without any processing.

Default

If none of these options is specified, the default option used is **SAVE**.

Optional

4. userid

Valid reply

This can be any valid Tivoli Information Management for z/OS user ID or application ID (APPLID) for an API user. This parameter is required if **NOTIFY** is specified; if any other other option is specified, this parameter must be omitted.

Default

None

Conditionally required.

5. stem

Valid reply

This is the stem name of a REXX compound variable containing the values to be inserted into the message.

Default

If a value is specified for **insertcount** but no value is specified for **stem**, a value of BLG_INSERT is used.

Optional

6. insertcount

Valid reply

This is the number of insert values included in the compound (**stem**) variable. If an insert value is specified in the **text** parameter, no value should be specified for the **insertcount** parameter.

Default

None

Optional

Usage Notes and Examples

There are three ways to specify message text:

- Panel name in **msgid**
- Message label in **msgid**
- Literal text in **text**

All TSX MESSAGE control lines return the complete message text in variable TSCAVDA. If any option other than **BUILD** is specified, other processing is also done.

This is an example of using a MESSAGE control line in a TSX to issue a message from a message panel with one inserted value. The message reason code is supplied by the text variable **retreascode**.

```
CALL BLGTSX 'MESSAGE','BLG9N003',retreascode,'SAVE';
```

This is an example of using a MESSAGE control line in a TSX to send a literal string message to another user. The contents of the text string is sent to the specified user ID. Note that the panel name is omitted.

```
CALL BLGTSX 'MESSAGE',,'I have assigned problem N990047 to you','NOTIFY',mapman;
```

This is an example of using a MESSAGE control line in a TSX to build BLGTMSGS CSECT message with label 10123, inserting VAL1, VAL2, and VAL3, and returning the message created to the TSX in TSCAVDA. When the TSX is run, the TSCAVDA gets set to `USR10123I This message has a first insert VAL1, a second insert VAL2, and a third insert VAL3.`

```
INS.1='VAL1'
INS.2='VAL2'
INS.3='VAL3'
CALL BLGTSX 'MESSAGE','10123',,'BUILD',,'INS.',3;
```

Additional considerations:

- If you pass the wrong number of inserts with a panel message, the message will contain blanks or the extras will be ignored.
- If you pass the wrong number of inserts with a CSECT message, ABEND U803 will result.
- Do not use 20xxx numbers in labels in BLGTMSGS; these are reserved for future product-supplied messages.

Return and Reason Codes

After the MESSAGE control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 21.

Table 21. MESSAGE Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0		Successful completion.
8	4	The message panel you specified cannot be loaded. Check to see if you specified the name correctly. If the name is correct, check to see if the message panel is in the read panel data set. If not, copy the message panel from your write panel data set. When you create the MESSAGE control line, Tivoli Information Management for z/OS does not check for the existence of the named message panel. If the name is correct and the panel exists, maybe you did not copy the panel to a read panel data set. All panels are initially filed in the write panel data set and must be copied to the read panel data set before processing. Also check that the session member includes the name of the read panel data set where the panel resides. If you still cannot resolve the problem, request a panel list of your read panel data sets and verify the panel name. Refer to the *Tivoli Information Management for z/OS Panel Modification Facility Guide* for information on listing panels.

Table 21. MESSAGE Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	8	The message panel you specified has no message text. Refer to the *Tivoli Information Management for z/OS Panel Modification Facility Guide* for instructions on creating a message panel.
8	12	The panel you specified is not a message panel. If you specify a penal, it must be a message panel. Verify that the name you entered on the MESSAGE Specification panel is the name of a message panel. If it is not, correct the name using panel update. If you are not sure of the name, request a panel list of your read panel data sets. Refer to the *Tivoli Information Management for z/OS Panel Modification Facility Guide* for information on listing panels.
8	16	An internal logic error occurred in National Language Support. Contact your Tivoli representative.
8	20	The user data is not a valid mixed string. Check the data and make the changes required to ensure that you specify valid mixed data.
8	24	The MESSAGE control line specified that variable data was to be used (**Get variable data** was set to YES), but the length of the data was 0. You must set up the variable data area and its length field before running a TSP containing a MESSAGE control line that specifies variable data. To set up the variable data area, call a user exit routine that sets the variable data area length and moves data into it or use the MOVEVAR control line. Check your TSP to make sure that a USEREXIT or MOVEVAR control line appears before the MESSAGE control line that caused this reason code. If a USEREXIT control line appears before the MESSAGE line, check that the routine sets the length field correctly.
8	28	The message you specified is too long. Rewrite it to be less than 254 characters.

You can use the TESTFIELD control line to test for these return and reason codes. See "TESTFIELD" on page 175 for information on how to do this.

TSCA Field Usage

Tivoli Information Management for z/OS sets the TSCA fields TSCAFRET (function return code) and TSCAFRES (function reason code) after a MESSAGE control line is run. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

If you do not enter **YES** in the **Save generated message** field for a TSP or else specify the **DISCARD** option for a TSX, Tivoli Information Management for z/OS increments the TSCA field TSCAMSGC (total messages).

- For a TSP:

If you enter **YES** in the **Get variable data** field and you set the variable data with a user exit routine, the user exit routine must set the length of the variable data in the TSCA field TSCAVDAL (current user variable data length).

If you set the variable data with the MOVEVAR control line, the MOVEVAR control line sets field TSCAVDAL for you.

If you specify a TSCA field that contains data in hexadecimal format, the data is *not* translated into EBCDIC for display. Attempts to display this data fail.

- For a TSX, the message is returned in variable TSCAVDA.

MOVEVAR

This control line enables you to add data to the variable data area of the TSCA control block. You can use the MOVEVAR control line to specify a string of data or a TSCA field that you want added to the TSCA control block. You can also use the MOVEVAR control line to replace data that is currently in the variable data area.

You can also use the MOVEVAR control line to store parameters in the variable data area (VDA) which can be passed to a terminal simulator EXEC (TSX) as one argument, which the TSX can then parse as needed. You can then use the TSP LINK control line to pass this argument to the TSX.

The function provided by the TSP MOVEVAR control line is available in a TSX using the standard program flow controls available in the REXX programming language.

Creating a MOVEVAR Control Line

Use the following MOVEVAR Specification panel to create a MOVEVAR control line:

```
BLM8CU9O              MOVEVAR SPECIFICATION                  PANEL:

Enter 'MOVEVAR' control data; cursor placement or input line entry allowed.



 1. TSCA Field Name.............. ________
 2. Literal Data................. ____________________________________
 3. Replace data................. NO_
 4. Data conversion.............. NO_


     When you finish, type END to save or CANCEL to discard any changes.









===>
```

Field Descriptions

1. TSCA Field Name

This field enables you to specify data to add to the variable data area.

Valid reply

The name of any existing TSCA field or no reply.

Default

No reply.

Restrictions

The length of the data to be moved is equal to the length of the TSCA field from which you want the data taken, with the exception of those fields that have associated length fields (such as TSCASDF and TSCACMRB). In the case of those fields, the value in the associated length field determines how much data to move.

If you enter no data in this field, enter no data in the **Literal Data** field, and set the **Replace data** field to **YES**, an SBCS blank is moved to the *beginning* of the variable data area.

If you enter no data in this field, enter no data in the **Literal Data** field, and set the **Replace data** field to **NO**, an SBCS blank is moved to the *end* of the variable data area.

If you enter data in this field and in the **Literal Data** field, the literal data is added to the variable data area first, followed by the value in this field.

2. Literal Data

This field enables you to specify literal data to be moved to the variable data area.

Valid reply

A string of 1 to 32 characters, or no reply.

Default

No reply.

Restrictions

When you require an SBCS comma as the first, or only, character of the **Literal data** field, you must precede the SBCS comma with an SBCS space character.

The data that is entered in this field is collected in the case entered by the user.

If you enter data in this field *and* in the **TSCA Field Name** field, the literal data is added to the variable data area first, followed by the TSCA field name.

If you enter no data in this field and no TSCA field name and the **Replace data** field is set to **YES**, an SBCS blank is moved to the *beginning* of the variable data area.

If you enter no data in this field and no TSCA field name and the **Replace data** field is set to **NO**, an SBCS blank is moved to the *end* of the existing variable data area.

3. Replace data

This field specifies whether you want the data being moved to replace any data that might already exist in the variable data area.

Valid reply

YES, NO, or no reply. A reply of **YES** in this field indicates that you want the existing data replaced. A reply of **NO** indicates that you want the data to be appended to any existing data.

Default

NO

4. Data conversion

This field specifies whether the data being moved is to be converted, and the type of conversion to be performed on it.

Valid reply

DEC, HEX, BIN, or **NO**.

A reply of DEC indicates that you want to convert binary data to printable decimal values and suppress leading zeros.

A reply of HEX indicates that you want to convert binary data to printable hexadecimal values and suppress leading zeros.

A reply of BIN indicates that you want to convert character data to binary data.

A reply of **NO** indicates that you do not want to convert data.

If you enter DEC or HEX in this field, leading zeros are removed before the data is moved. If the data is 0, a single 0 is moved.

Default

NO

Restrictions

If the data conversion type is HEX or DEC, the data can be no longer than 4 bytes. If the conversion type is BIN, the data can be no longer than 8 bytes.

Usage Notes

Depending on the data you specify, the variable data area can contain new or appended data whose source is either a TSCA field, user-specified literal data, or both. The length of the variable data area is 512 bytes.

For examples of how to use this control line, use PMF to look at TSPs BTNTAC1R and BTNTAPRV in the base panel data set.

What the Control Line Does

When the TSP is run, Tivoli Information Management for z/OS extracts the data to be moved to the variable data area from either the control line or from the specified TSCA field. The extracted data is then added to the end of the existing data in the variable data area if you put **NO** in the **Replace data** field. If you put **YES** in the **Replace data** field, the extracted data is placed at the beginning of the buffer. The length of the used part of the variable data area (TSCAVDAL) is calculated.

Return and Reason Codes

After a MOVEVAR control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 22 on page 146.

Table 22. MOVEVAR Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion.
	4	The **Replace data** field was specified as YES, but there was no data to replace. The requested data was moved.
8	4	The length of the data to be moved is not valid. No data was moved, and the contents of the variable data area remain unchanged. If the data conversion choice is BIN, the input must be 8 bytes or fewer. If the data conversion is HEX or DEC, the input must be 4 bytes or fewer.
8	8	The data to be moved is longer than 512 characters and is too long to fit in the variable data area. No data was moved, and the contents of the variable data area remain unchanged.
8	12	The data conversion to binary was not performed because the input data is not numeric. No data was moved.

You can use the TESTFIELD control line to test for these return and reason codes. See "TESTFIELD" on page 175 for information on how to do this.

TSCA Field Usage

The following TSCA fields are set by Tivoli Information Management for z/OS when processing a MOVEVAR control line. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

TSCAFRET
: Function return code

TSCAFRES
: Function reason code

TSCAVDAL
: Variable data area length and the buffer pointed to by TSCAVDAP.

OPENRRES

This Remote Data Resource TSX control line is described in "OPENRRES" on page 244.

OPENSOCKET

This control line opens a TCP/IP socket and establishes a connection with a waiting server. It uses the assembler callable services to invoke the OS/390 UNIX System Services version of the TCP/IP product installed. If a callable service does not complete successfully, the name of the service called is returned in the NETFUNC REXX variable, the return code is returned in the NETRETC REXX variable, and the reason code is returned in the NETREAC REXX variable. The user should refer to the OS/390 UNIX System Services Messages and Codes manual and the OS/390 UNIX System Services Assembler Callable Services manual for a description of the returned values. OPENSOCKET also sets the value 8 in TSCAFRET.

More than one TCP/IP connection can be established with a TSX concurrently. A socket identifier is returned from OPENSOCKET and is used to specify the connection to be accessed. When the TSX ends, any TCP/IP connections remaining open are closed.

This control line can be called only from a TSX. An ISPF or TSO environment is not required in order to use OPENSOCKET.

The OPENSOCKET Control Line

The format of the OPENSOCKET control line is:

```
CALL BLGTSX 'OPENSOCKET',ipaddress,portnumber
```

Parameter Descriptions

1. ipaddress

Valid reply

The internet or IP address of the host server to be connected. It is specified in dotted decimal format. This consists of four numbers with valid values from 0 to 255, separated by periods.

Default

None

Required

2. portnumber

Valid reply

The port number of the host server to be connected. It is specified by a single integer; the value of this integer can be any value of 1 to 65535.

Default

None

Required

Usage Notes and Examples

This is an example of using an OPENSOCKET control line in a TSX.

```
CALL BLGTSX 'OPENSOCKET','9.18.153.2','1234'
```

When the control line completes successfully, the NETSOCKET REXX variable contains the socket ID of the TCP/IP connection just established. It should be saved in a variable and used as an input parameter to the WRITESOCKET, READSOCKET, and CLOSESOCKET control lines.

Note: Six REXX variables have been defined to return information from the OPENSOCKET control line. These variables (NETSOCKET, NETDATA, NETBYTECOUNT, NETFUNC, NETRETC, and NETREAC) are reset during the processing of OPENSOCKET. It is the responsibility of the TSX to save any data needed for processing.

Return and Reason Codes

After the OPENSOCKET control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 23.

Table 23. OPENSOCKET Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion.

Table 23. OPENSOCKET Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	4	The TCP/IP service did not complete successfully. Refer to the OS/390 UNIX System Services Messages and Codes manual and the OS/390 UNIX System Services Assembler Callable Services manual for a description of the following return codes. These REXX variables contain the diagnostic information: **NETFUNC** The name of the Assembler Callable Service being invoked. **NETRETC** The value of the Return_code parameter returned by the service. **NETREAC** The value of the Reason_code parameter returned by the service.

PRINT

This control line enables you to print the following kinds of information:

- Messages (saved and current)
- Current panel that would have been displayed in the corresponding interactive session
- Contents of the TSCA (in hexadecimal) through the residual reply buffer (TSCARRB).

Refer to "The PRINT TSX Control Line" on page 151 for information about using the PRINT control line in a TSX.

If you want to see what messages have been generated by a portion of your TSP, enter a PRINT control line. Using a PRINT control line in an error routine can help you determine why your TSP branched to the error routine.

This example uses a PRINT control line to print messages before data is sent to Tivoli Information Management for z/OS by a PROCESS control line.

```
PROCESS
⋮

ADDDATA
PRINT             (Print messages generated from the previous
                   PROCESS control line and any saved messages)
PROCESS
ADDDATA
USEREXIT
PRINT             (Print messages generated from the previous
                   PROCESS control line, USEREXIT control line,
                   and any saved messages)
```

```
PROCESS
:
```

Creating a PRINT Control Line

Use the following PRINT Specification panel to create a PRINT control line:

```
BLM8CU9R                PRINT  SPECIFICATION                PANEL: ________

Enter 'PRINT' control data; cursor placement or input line entry allowed.



                  1. Print the messages........ YES
                  2. Print the screen.......... NO_
                  3. Print the TSCA............ NO_




     When you finish, type END to save or CANCEL to discard any changes.









===>
```

Field Descriptions

1. Print the messages

Valid reply

YES, NO, or no reply.

If you enter **YES** in this field, a copy of the current message chain and the saved message chain are written to the output destination you defined on the SYSPRINT DD statement. Any other valid reply results in the messages not being printed.

Default

YES

Restrictions

None

2. Print the screen

Valid reply

YES, NO, or no reply.

If you enter **YES** in this field, a copy of the current panel is printed to the output destination you defined on the SYSPRINT DD statement. Any other valid reply results in the current panel not being printed.

Default

NO

Restrictions
None

3. Print the TSCA

Valid reply
YES, NO, or no reply.

If you enter **YES** in this field, a copy of the current TSCA is printed to the output destination you defined on the SYSPRINT DD statement. Any other valid reply results in the TSCA not being printed.

Default
NO

Restrictions
None.

Usage Notes

Use the PRINT control line in your error routines to help you determine the cause of errors. For more information on using PRINT to test your TSPs, see "Using the PRINT Control Line" on page 251. Also, for examples, use PMF to look at TSPs BTNTARPP and BTNTARUP in your base panel data set.

If Print the messages is **YES**, current and saved messages are printed. Current messages are messages created since the last PROCESS or MESSAGE control line ran. Saved messages are messages that:

- Existed on the message chain where the TSP is started
- Are saved by using the **Save existing messages** option on a Process control line
- Are saved by using **Save generated message** on a MESSAGE control line.

For more information about TSP messages, see "Message Handling during TSP and TSX Processing" on page 261.

What the Control Line Does

When the TSP is run, the information that is printed is sent to the destination you defined on the SYSPRINT DD statement, to the SYSOUT device, or to another output data set. If you direct the output to another data set, the DCB information for the output data set must include RECFM=VBA.The LRECL default is set to 137. You can change this length when you define the data set. However, if an output line is longer than the specified LRECL, it is split across as many print lines as necessary to print the line.

You must allocate the SYSPRINT DD statement before calling the TSP, whether a DD statement processed at the time Tivoli Information Management for z/OS is called or ISPF 'TSO ALLOCATE...' is issued during a Tivoli Information Management for z/OS session.

When Tivoli Information Management for z/OS writes to SYSPRINT, it formats the data using DCB information that was specified on either a SYSPRINT DD statement (that is, LRECL or BLKSIZE) or a TSO ALLOCATE statement. If LRECL was specified without BLKSIZE, Tivoli Information Management for z/OS sets BLKSIZE to:

```
(14 * LRECL) + 4
```

If neither BLKSIZE or LRECL, was specified, LRECL is set to:

```
(length of output message) + 4
```

and BLKSIZE is set to:

```
(14 * LRECL) + 4
```

If BLKSIZE is specified without LRECL, LRECL is set to the smaller of the following statements:

```
(length of output message) + 4
```

or

```
 BLKSIZE - 4
```

In all cases, LRECL must be less than or equal to (BLKSIZE - 4). If this is not the case, an ABEND occurs when the data set is opened, because the data attributes are inconsistent.

Refer to the *TSO Extensions V2 CLISTs* manual for additional information on allocating data sets.

The PRINT TSX Control Line

The format of the PRINT control line is:

```
CALL BLGTSX 'PRINT',item1,item2,item3
```

Parameter Descriptions

1. item1, item2, and item3

Valid reply

The type of information to be printed.

MESSAGES

Print the messages on the current message chain and the saved message chain.

SCREEN

Print a copy of the current screen (panel).

TSCA Print the contents of the current TSCA.

The values can be specified in any order.

Default

None

Required/Optional

Item1 is required. Item2 and item3 are optional.

Usage Notes and Examples

This is an example of using a PRINT control line in a TSX. The current messages, screen, and the contents of the TSCA will be printed.

```
CALL BLGTSX 'PRINT','MESSAGES','SCREEN','TSCA';
```

Return and Reason Codes

After a PRINT control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 24 on page 152.

Table 24. PRINT Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0		All requested areas were printed.
8	4	An internal logic error occurred. Nothing was printed. Contact your Tivoli representative.
8	8	Data set attributes are not valid for the output device. See "What the Control Line Does" on page 150 for the correct data set attributes.
8	12	Permanent I/O error, or data set is full. Define a larger print data set. Nothing was printed.

You can use the TESTFIELD control line to test for these return and reason codes. See "TESTFIELD" on page 175 for information on how to do this.

TSCA Field Usage

After a PRINT control line is run, Tivoli Information Management for z/OS sets the following TSCA fields. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

TSCAFRET
Function return code

TSCAFRES
Function reason code.

PROCESS

In a TSP, this control line acts as an Enter key, sending the responses accumulated in the TSCA command line reply buffer (TSCACMRB) to Tivoli Information Management for z/OS for processing. In a TSX the responses are passed as a parameter on the PROCESS control line.

You can use a PROCESS control line with no data in the command line reply buffer to simulate a null reply in a TSP. Similarly, a PROCESS control line in a TSX that is not passed the 'response parameter' will also simulate a null reply.

Refer to "The PROCESS TSX Control Line" on page 155 for information about using the PROCESS control line in a TSX.

This example uses a PROCESS control line to send data to Tivoli Information Management for z/OS:

```
ADDDATA    3,2,6,1,SE + PERA/SMITH
PROCESS    ERROR     (sends above IRC to Tivoli Information Management for z/OS for processing.
                       if there is a problem, branch to the ERROR label)
⋮

LABEL      ERROR
```

Creating a PROCESS Control Line

Use the following PROCESS Specification panel to create a PROCESS control line:

```
BLM8CU9H                PROCESS SPECIFICATION                PANEL: ________

Enter 'PROCESS' control data; cursor placement or input line entry allowed.



                1. Error label name.........<R> ERROR___
                2. Save existing messages?......NO_









     When you finish, type END to save or CANCEL to discard any changes.









===>
```

Field Descriptions

1. Error label name

This field indicates the name of a label that identifies a control line in the current TSP where processing resumes if the return code for the PROCESS control line is 8. This field is required.

Valid reply

A mixed string of 1 to 8 characters, beginning with an SBCS alphanumeric or national character or an SO character.

Default

None.

Restrictions

The label name you enter in this field must be a valid name that is identified through the LABEL control line. Also, the name must exist in the same TSP as the PROCESS control line.

2. Save Existing Messages?

This field is used to specify what happens to any messages that might exist on the current message chain when the PROCESS function is started.

Valid reply

A value of **YES** in this field indicates that messages on the current message chain are to be saved. These messages are added to the *saved message chain*. The saved message chain might also contain messages that already existed there at the time the TSP was started. If your TSP contains MESSAGE control lines with the **Save generated message** field set to **YES**, any generated messages also exist on the saved message chain.

A value of **NO** in this field indicates that messages on the current message chain are to be deleted.

You can use TESTFLOW to query messages on either the current or saved message chain. Use the return code and reason code combination (4/4) to verify messages on the saved message chain.

Default
NO

Usage Notes

Entering a PROCESS control line with **NO** in the **Save existing messages** field and nothing in the reply buffer is like entering a null reply. You can use this method to clear messages generated since the TSP was started. For example, if your TSP is updating records from a large search results list, your TSP ends with the *Record filed successfully* message repeated many times. To avoid this situation, do a TESTFLOW for that message after each PROCESS. If the message is located, do another PROCESS to clear the message.

Note: When entering dates and times, be sure to pass them in the external format which the user is currently using. Use BLGEDATE to convert an internal date to external date format. The user exit BLGEDATE is described in "General-purpose User Exits" on page 279.

For examples, use PMF to look at TSPs BTNTACLS and BTNTAC1R in your base panel data set.

It is recommended that you check for return codes before processing another control line (such as TESTFLOW) that resets the return and reason codes.

What the Control Line Does

If the return code for the PROCESS control line is 0 or 8, the data has been successfully transmitted to Tivoli Information Management for z/OS for processing. Upon return, the command line reply buffer is cleared. When an error is found before the data in the command line reply buffer is sent, the command line reply buffer is not cleared, and processing continues at the next control line. If the return code is 8 after the command line reply buffer is processed, a branch is made to the error target you specified on the PROCESS Specification panel.

When this control line is run, the TSCAMSGC field is first set to 0, regardless of the setting in the **Save existing messages** field. If the processing of the control line generates more messages, the TSCAMSGC field is incremented as needed. TSCAMSGC is reset even if messages were previously saved on the chain.

Table Panel Processing

There are restrictions on processing a table panel with a TSP:

- The table panel must be one that you can update.
- Unless it is used by the list processor (program exit BLG01396), the table panel must contain only one input data area, excluding the line command area. If the table panel contains no input data area (such as a search results list), or if it contains more than one input data area (such as a dictionary update panel) and is not a list processor table panel, the table is not processed.

- A maximum of 32 767 lines can be displayed in table panels (including search results lists). Your TSP could exceed this maximum if it requests a search that returns a large number of records found. If the number of records found is greater than this maximum, the search results list contains only that maximum number of records that can be displayed. The actual number of records found appears in the upper right corner of the search results list.

  Because this maximum value is customizable, the maximum for your installation could be less than 32 767. This value is set during installation in the SORTPFX session parameter. For more information about SORTPFX, refer to the *Tivoli Information Management for z/OS Planning and Installation Guide and Reference*.

If the table panel meets these criteria, Tivoli Information Management for z/OS copies the data into the TSCA field TSCATBLL (current table display line) and sets the TSCACTBL (current table panel line length), TSCATPLC (total line count), TSCATPLN (current line number), and TSCAMTBL (maximum table panel line length) fields. If the table panel does not meet these criteria, no data is copied, and TSCAMTBL is set to zero.

You must write a user exit routine that manipulates the data in the current table display line and sets the current line length to the correct value. If the table panel is used by the list processor, you can manipulate the fields using the line entry line command.

When the PROCESS control line is run, the data in TSCATBLL replaces the contents of the current line of the table panel. Any data in the command line reply buffer is processed after the table panel's current line is updated.

You update the table panel lines by coding a combination of USEREXIT, ADDDATA, and PROCESS control lines in your TSP. See "Examples: Adding or Updating Freeform Text" on page 49 for more information about updating table panel lines.

The PROCESS TSX Control Line

This control line performs the TSP functions ADDDATA and PROCESS. Instead of collecting the responses in the TSCA field TSCACMRB (command line reply buffer), the responses are passed as a parameter on the invocation of the TSX PROCESS control line.

The format of the PROCESS control line is:

```
CALL BLGTSX 'PROCESS',responses,save
```

Parameter Descriptions

1. responses

Valid reply

The response string to be processed up to 512 bytes.

Default

None

Optional

2. save

Valid reply

Indicates whether messages on the current message chain are to be saved or discarded.

SAVE Save the messages on the current message chain.

DISCARD
Discard the messages on the current message chain.

Default
DISCARD

Optional

Usage Notes and Examples

This example shows an example of using a PROCESS control line in a TSX. The record with RNID USERS is displayed.

```
CALL BLGTSX 'PROCESS',';DISPLAY R USERS','SAVE';
```

Upon completion of the TSX PROCESS control line, any messages currently on the message chain will be in the REXX compound variable BLG_MESSAGE, with the count of messages (TSCAMSGC) contained in variable BLG_MESSAGE.0.

Return and Reason Codes

After a PROCESS control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 25.

Since the TSX PROCESS control line performs both TSP ADDDATA and PROCESS functions, you will need to refer to both the TSP ADDDATA and PROCESS return and reason codes. If the TSXPROCESS TSCAFRES reason code has a 1000 added to it, refer to the ADDDATA return and reason codes. Otherwise, refer to the PROCESS return and reason codes.

Table 25. PROCESS Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	The reply was successfully processed. No messages were received.
0	4	The reply was successfully processed. Informational messages were received.
4	4	Tivoli Information Management for z/OS found errors before it could process the responses in the command line reply buffer. The buffer was not cleared. The data length of the current table panel line (TSCATBLL) is greater than the maximum allowed (TSCAMTBL). Processing continues at the next sequential control line. Your TSP attempted to put too many characters in the current table panel line. Your user exit routine needs to check the maximum length of TSCA field TSCAMTBL before entering data. Printing the TSCA will help you resolve the problem.

Table 25. PROCESS Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	8	Tivoli Information Management for z/OS generated warning messages. Any residual data that was not processed by Tivoli Information Management for z/OS is returned in the TSCA residual-reply buffer. If the TSP is running under an API session, a severe error might have occurred and messages have been issued. If a severe error occurred, the command line reply buffer is cleared, and the current dialog is ended. The severe error message panel does not appear. If you anticipate warning messages at this point, you can design your TSP to recover from them. If not, you can update your TSP to go to an error routine and print the messages before returning.
8	12	Tivoli Information Management for z/OS detected a severe error and issued one or more messages. Any residual data that was not processed by Tivoli Information Management for z/OS is returned in the TSCA residual-reply buffer. When this occurs, the only valid Tivoli Information Management for z/OS responses are BACK, CANCEL, INIT, QUIT, and HELP. You can design your TSP to go to an error routine before ending. The error routine could have the following sequence of control lines: PRINT (panel, messages), ADDDATA (;HELP), PROCESS, PRINT (panel). This would print the HELP messages.

You can use the TESTFIELD control line to test for these return and reason codes. See "TESTFIELD" on page 175 for information on how to do this.

TSCA Field Usage

Tivoli Information Management for z/OS sets the following TSCA fields after a PROCESS control line is run. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

TSCAFRET
: Function return code

TSCAFRES
: Function reason code

TSCACMOF
: Offset in command line reply buffer

TSCARRB
: Residual reply buffer

TSCARRBL
: Length of residual reply buffer

TSCATPLC
: Lines in the current table panel

TSCATPLN
: Current line number of table panel

TSCAMTBL
: Maximum data length for current table panel line

TSCACTBL
Current length of data in current table panel line
TSCATBLL
Current table panel line
TSCAMSGC
Total messages.

PUTRDATA

This Remote Data Resource TSX control line is described in "PUTRDATA" on page 245.

QMAIL

This control line is used to send a constructed notification mail message to a BLX-SP.

This control line can be called only from a TSX. Also, it is a special purpose control line for handling mail and is not generally useful for writing TSXs.

The QMAIL Control Line

The format of the QMAIL control line is:

```
CALL BLGTSX 'QMAIL',numberoflines,queuename,maxlinelength,stemname
```

Parameter Descriptions

1. numberoflines

Valid reply
The number of the elements in the input compound variable.

Default
None

Required

2. queuename

Valid reply
The name of the BLX-SP queue to put the mail on.

Default
MAILQ1

Optional

3. maxlinelength

Valid reply
The length of the longest line in the message. Lines are read from the compound variable. If a line is longer than the specified maximum length, it is truncated.

Valid values can be between 1 and 255 inclusive. If the value is greater than 255, the system sets it to 255. If the value is 0, it is set to 1.

Default
80

Optional

4. stemname

Valid reply
The name of a REXX compound variable (including a separator character, such as a period) that contains the mail message.

Default
BLG_QMAIL.

Optional

Usage Notes and Examples

This example shows how to use a QMAIL control line in a TSX. Mail is placed on the default queue.

```
CALL BLGTSX 'QMAIL',BLG_QMAIL.0;
```

This is another example, showing how to use a QMAIL control line in a TSX. Mail is placed on the default queue.

```
CALL BLGTSX 'QMAIL',MSG.0,,,'MSG.';
```

Return and Reason Codes

After the QMAIL control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 26.

Table 26. QMAIL Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Processing successful. The mail is queued to the BLX-SP.
0	4	Processing successful. The mail is mail queued to the BLX-SP. However, the warning limit set by either the MAILQ BLXPRM keyword or the MAILQ operator command was exceeded.
8	4	Processing unsuccessful. The mail is not queued to the BLX-SP. The maximum limit set by either the MAILQ BLXPRM keyword or the MAILQ operator command was exceeded.
8	8	Processing unsuccessful. The mail is not queued to the BLX-SP. Unable to obtain storage in the BLX-SP to hold the new mail item.
12	4	Processing unsuccessful. The mail is not queued to the BLX-SP. The queue has been closed.
12	8	Processing unsuccessful. The mail is not queued to the BLX-SP. The queue name specified on the call to BLGTSX does not exist.

QUERYRRES

This Remote Data Resource TSX control line is described in “QUERYRRES” on page 246.

READDICT

This control line enables you to read the dictionary. Typically, you use the results from this control line as input to another control line. For example, the TSX GETLIST control line requires an s-word. You can use the READDICT control line to get the s-word using the s-word index.

This control line can be called only from a TSX.

The READDICT Control Line

The format of the READDICT control line is:

```
CALL BLGTSX 'READDICT',index
```

Parameter Descriptions

1. index

Valid reply

The index of the dictionary entry to be read. Valid values are an S (for s-words) or P (for p-words) followed by four hexadecimal characters (for example, S0BEE). If an s-word if read, it is stored in TSCARSD; if a p-word is read, the prefix (if any) is stored in TSCARPD and the validation (if any) is stored in TSCASDF.

Default

None

Required

Usage Notes and Examples

This example uses a READDICT control line in a TSX. The s-word for s-word index X'0BEE' is stored in TSCARSD.

```
CALL BLGTSX 'READDICT','S0BEE'; /* GET S-WORD FOR STATUS */
```

Return and Reason Codes

After the READDICT control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 27.

Table 27. READDICT Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Processing successful.
0	4	Processing successful; dictionary entry is blank.
8	8	Processing unsuccessful; dictionary entry not found or could not be read.

TSCA Field Usage

The following TSCA fields are set by Tivoli Information Management for z/OS when an READDICT control line is processed. For more information about these fields, see “Terminal Simulator Communications Fields” on page 289.

TSCAFRET

Function return code.

TSCAFRES

Function reason code.

TSCARSD
S-word associated with the specified s-word index.

TSCARPD
P-word associated with the specified p-word index.

TSCASDF
Data associated with the specified p-word index.

READSOCKET

This control line receives data sent from a server over a previously opened TCP/IP connection. This is a non-blocking operation, so that control is returned to the caller of READSOCKET immediately after the data has been given to TCP/IP. The NETBYTECOUNT field gives the length of the data returned by the TCP/IP service in the NETDATA field. The length value may be from zero up to the datalen value specified on the READSOCKET invocation. If a callable service does not complete successfully, the name of the service called is returned in the NETFUNC REXX variable, the return code is returned in the NETRETC REXX variable, and the reason code is returned in the NETREAC REXX variable. The user should refer to the OS/390 UNIX System Services Messages and Codes manual and the OS/390 UNIX System Services Assembler Callable Services manual for a description of the returned values.

This control line can be called only from a TSX. An ISPF or TSO environment is not required in order to use READSOCKET.

The READSOCKET Control Line

The format of the READSOCKET control line is:

```
CALL BLGTSX 'READSOCKET',socketid,datalen
```

Parameter Descriptions

1. socketid

Valid reply
The socket identification for this TCP/IP connection. This value was initially returned from the OPENSOCKET control line in the NETSOCKET REXX variable.

Default
None

Required

2. datalen

Valid reply
The length of the data to be received. The value specified must be greater than 0.

Default
None

Required

Usage Notes and Examples

This is an example of using a READSOCKET control line to read data from a server. When the control line completes successfully, the NETBYTECOUNT REXX variable contains the count of the number of bytes received. If this count is less than the amount expected, READSOCKET should be invoked in a loop until the total bytes expected are received. The data read is returned in the NETDATA REXX variable.

Note: Six REXX variables have been defined to return information from the READSOCKET control line. These variables (NETSOCKET, NETDATA, NETBYTECOUNT, NETFUNC, NETRETC, and NETREAC) are reset during the processing of READSOCKET. It is the responsibility of the TSX to save any data needed for processing.

```
Expected = 10
Received = 0
Data = ''
DO WHILE Expected > Received
  CALL BLGTSX 'READSOCKET',SaveSocket,Expected-Received
  IF NETBYTECOUNT > 0 THEN
     Data = Data NETDATA
  Received = Received + NETBYTECOUNT
END
```

Return and Reason Codes

After the READSOCKET control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 28.

Table 28. READSOCKET Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion.
8	4	The TCP/IP service did not complete successfully. Refer to the OS/390 UNIX System Services Messages and Codes manual and the OS/390 UNIX System Services Assembler Callable Services manual for a description of the following return codes. These REXX variables contain the diagnostic information: **NETFUNC** The name of the Assembler Callable Service being invoked. **NETRETC** The value of the Return_code parameter returned by the service. **NETREAC** The value of the Reason_code parameter returned by the service.

RELEASERRES

This Remote Data Resource TSX control line is described in “RELEASERRES” on page 247.

REPLIST

This control line enables you to replace one line or a block of lines in a list processor list.

This control line can be called only from a TSX.

The REPLIST Control Line

The format of the REPLIST control line is:

```
CALL BLGTSX 'REPLIST',listsword,stemname,startln,oldcount,newcount
```

Parameter Descriptions

1. listsword

Valid reply

The root s-word of the list in which items are to be replaced. The root s-word includes the hexadecimal watermark character; therefore, it is recommended that you use the TSX READDICT control line to get the root s-word that you will pass to REPLIST. Refer to "READDICT" on page 160 for more information on the READDICT control line.

Default

None

Required

2. stemname

Valid reply

The name of a REXX compound variable (including a separator character, such as a period) for the compound variable containing the new items.

Default

BLG_LIST.

Optional

3. startln

Valid reply

The line number of the first line to be replaced. Valid values are 1 to 19274.

Default

None

Required

4. oldcount

Valid reply

The number of items to be removed from the original text. Valid values are 0 to 19274 or ALL. If 0 is specified, REPLIST simply inserts items prior to the line specified by *startln*. If ALL is specified, REPLIST replaces all lines from *startln* to the end of the list.

Default

1

Optional

5. newcount

Valid reply

The number of new items. Valid values are 0 to 19274. If 0 is specified, REPLIST replaces the specified block of lines with nothing, which is the same function as a DELETE.

Default

The value of *oldcount*.

Required if *ALL* is specified for *oldcount*; if *ALL* is not specified for *oldcount*, then this parameter is Optional.

CAUTION:
The list processor data that the TSX REPLIST control line retrieves must be in the internal format introduced in Version 5.1 (or in Version 4 by APARs OY47188 and OY47893). Lists which were stored prior to the internal format change will not be retrieved accurately by REPLIST. Those lists can be converted to the new internal format by updating them in Version 7.1, repeating the first line, deleting the first line, and filing the record. The data will be unchanged, but it will be stored in the correct format.

Usage Notes and Examples

This is an example of using a REPLIST control line in a TSX to replace the fifth item in the device name list.

```
index='S1416';
CALL BLGTSX 'READDICT',index;   /* Get the s-word.                        */
sword=TSCARSD;                  /* The s-word of the device name list     */
moditem.1='DEVICE7'
CALL BLGTSX 'REPLIST',sword,'moditem.',5
```

Return and Reason Codes

After the REPLIST control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 29.

Table 29. REPLIST Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	The specified items were replaced.
0	4	The specified items were replaced; the specified range extended beyond the end of the list.
0	8	The specified range was beyond the end of the list; the new items were added, with blank lines added, if necessary, between the end of the previous list and the beginning of the new items.
8	4	The function could not be completed because it would cause the list to exceed the maximum allowed size (19274) for a list; items were added up to line 19274.
8	8	The value of a list item failed validation checking. The list is updated and the specified number of lines are replaced, but the data for the line that failed validation and all subsequent lines is set to blanks. TSCATLIX contains the index of the first item which failed validation.
8	12	No list of the specified type existed; no new list items were added to the record. Use the ADDLIST TSX control line to add new list items.
12	4	The control line function failed. This control line cannot be run while the list processor is active.

REPTEXT

This control line enables you to replace an existing block of text with new lines of text.

This control line can be called only from a TSX.

The REPTEXT Control Line

The format of the REPTEXT control line is:

```
CALL BLGTSX 'REPTEXT',sword,stemname,startln,oldcount,newcount
```

Parameter Descriptions

1. sword

Valid reply

The structured word associated with the text to be replaced.

Default

None

Required

2. stemname

Valid reply

The stem name (including any separator character such as a period) for the compound variable containing the new text.

Default

BLG_TEXT.

Optional. If this parameter is omitted, the stem BLG_TEXT. is used.

3. startln

Valid reply

The line number of the first line to be replaced. Valid values are 1 to 999999.

Default

None

Required

4. oldcount

Valid reply

The number of lines to be removed from the original text. Valid values are 0 to 99999 or ALL. When 0 is specified, REPTEXT simply inserts text prior to the line specified by *firstln*. When ALL is specified, REPTEXT replaces all lines from *startln* to the end of the file.

Default

1

Optional

5. newcount

Valid reply

The number of new text lines. Valid values are 0 to 99999. When 0 is specified, REPTEXT replaces the specified block of text with nothing, which is the same as a DELETE function.

Default

1

Required if *ALL* is specified for *oldcount*. If omitted, the value specified for *oldcount* is used.

Usage Notes and Examples

Following is an example of using a REPTEXT control line in a TSX to replace the fifth line of problem description text.

```
index='S0E01';
CALL BLGTSX 'READDICT',index;             /* Get the s-word.              */
sword=TSCARSD;                            /* The s-word of the text data. */
modtext.1='Modified text line'
CALL BLGTSX 'REPTEXT',sword,'modtext.',5
```

This example uses a REPTEXT control line in a TSX to replace lines 7 through 9 of problem description text with a single line of modified text.

```
index='S0E01';
CALL BLGTSX 'READDICT',index;              /* Get the s-word.              */
sword=TSCARSD;                             /* The s-word of the text data. */
modtext.1='Modified text line'
/* Replace 3 lines starting at line 7 with one modified line   */
CALL BLGTSX 'REPTEXT',sword,'modtext.',7,3,1
```

Note: You can use the REXX STRIP() function to remove any trailing blanks that might exist in the text prior to processing the REPTEXT control line.

Return and Reason Codes

After the REPTEXT control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 30.

Note: If you are using the Tivoli Information Management for z/OS freeform text editor, you should be aware that this editor adds additional blank lines to a record in order to present a full screen for editing. As a result, the reason code will, in some cases, be different according to whether you do REPTEXT from within the editor or do REPTEXT when you are not in the editor.

Table 30. REPTEXT Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	The specified text was replaced.
0	4	The specified text was replaced; the specified range extended beyond the end of the text.
0	8	The specified text range was beyond the end of the text; the new text was added, with one or more blank lines added between the end of the previous text and the beginning of the new text.

Table 30. REPTEXT Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	12	No text of the specified type existed; the new lines were added in the specified range with blank lines added, if necessary, prior to the beginning of the new lines.
8	4	The length of one or more input text lines exceeds the maximum allowable line length of 132. No text is updated.

RETURN

This control line provides an exit from a TSP. The REXX ″exit″ statement provides equivalent function for a TSX. However, any information returned is ignored.

This example uses a RETURN control line to exit the TSP when processing completes.

```
PROCESS     ERROR
LABEL       DONE
⋮

RETURN                              (Return at this point to avoid
                                     processing your error routine)
LABEL       ERROR
```

Creating a RETURN Control Line

Because this control line does not collect any input data, it does not have a specification panel. After you create the RETURN control line from the Function Name panel, you are returned to the updated Function Line Summary panel.

Usage Notes

Although not necessary, it is helpful if you use a RETURN control line at the end of each TSP. The one exception to this is when you run a TSP in the batch environment. Using an ADDDATA control line with ;QUIT in the **Literal data** field will cause the TSP to exit Tivoli Information Management for z/OS when it completes. In this way you can avoid unnecessary messages.

What the Control Line Does

If the current TSP received control because a previous TSP issued a LINK control line, processing resumes with the control line following the LINK control line in the previous TSP.

If this TSP was started by the 002B function code, the 001B function code, the RUN command, or the TSP parameter, processing returns to the point in the Tivoli Information Management for z/OS environment where the TSP was when it ended its processing.

Return and Reason Codes

After a RETURN control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to zero. (See Table 31 on page 168.)

Table 31. RETURN Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion.

SETAPIDATA

This control line is used to return information to an application that invoked the High-Level Application Program Interface (HLAPI). In other words, SETAPIDATA provides a means of using a TSX to create an output parameter data block (PDB) for a HLAPI application. The HL14 (Start User TSP) transaction is issued to explicitly invoke a TSP or TSX. TSXs invoked during the processing of the HL05 (Check In Record), HL08 (Create Record), and HL09 (Update Record) transactions can also return data as output PDBs using SETAPIDATA. These output PDBs will follow any output PDBs generated by HLAPI processing. Each call to SETAPIDATA causes an output PDB to be constructed and chained to any previously built output PDBs. LLAPI applications using the T111 (Start User TSP) transaction will not have access to the returned data. The *Tivoli Information Management for z/OS Application Program Interface Guide* contains additional information about the HLAPI and PDBs.

If SETAPIDATA receives a single string of data, the length of the data must be less than 32767. If SETAPIDATA receive a stem variable, the length of each line in the stem must be less than 255.

This control line can be called only from a TSX and is valid only in an API environment.

The SETAPIDATA Control Line

The format of the SETAPIDATA control line is:

```
CALL BLGTSX 'SETAPIDATA',pdbname,data,numberoflines,maxlinelength
```

Parameter Descriptions

1. pdbname

Valid reply

The name to assign to the resulting output PDB. Maximum length is 32 characters.

Default

None

Required

2. data (or stemname)

Valid reply

If numberoflines is not specified, this value is put into PDBDATA and the PDBDATW value is set to 0. If numberoflines is specified, this value is treated as the stem of a REXX compound variable that will provide the data for PDBDATA. Data from each segment of the compound variable will be truncated or padded with blanks to the length of maxlinelength and PDBDATW is set to the maxlinelength.

Default

None

Required

3. numberoflines

Valid reply

The number of elements in the input compound variable. If specified, the value specified in the data or stemname parameter is treated as a compound variable stem. If not specified, the value specified in the data or stemname parameter is treated as a string.

Default

None

Optional

If specified, maxlinelength is required.

4. maxlinelength

Valid reply

The length of the longest line in the specified compound variable. This value cannot be greater than 255. Lines are read from the compound variable. If a line is longer than the specified maximum length, it is truncated. If a line is shorter than the specified maximum length, it is padded with blanks when put into PDBDATA. The PDBDATW of the resulting output PDB is set to this maxlinelength.

Default

None

Optional

If specified, numberoflines must be specified.

Usage Notes and Examples

This is a simple example of using an SETAPIDATA control line in a TSX.

```
CALL BLGTSX 'SETAPIDATA','OUTPUT1','DATA1'
```

This is a slightly more complex example of using the SETAPIDATA control line in a TSX.

```
STEM2.1='First line of output data'
STEM2.2='Second line of output data'
STEM2.3='This is the last line of output data'
CALL BLGTSX 'SETAPIDATA','OUTPUT2','STEM2',3,36
```

Return and Reason Codes

After the SETAPIDATA control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 32.

Table 32. SETAPIDATA Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Processing successful. An element is constructed that the HLAPI will use to build an output PDB.
8	4	API not active.

SETFIELD

This control line enables you to set a field (TSCAUFLD or TSCATLIX) in the TSCA that you can use for communication within a TSP or between several TSPs or user routines.

The TSP SETFIELD function is available in a TSX using standard program operators available in the REXX programming language. Although you cannot directly set TSCA fields in a TSX, TSX control lines provide parameters that allow you to pass information necessary to complete the control line functions. Therefore, you do not need the SETFIELD control line a TSX.

You can use this control line in several ways. For instance, you might have a TSP that is called by several TSPs. You want the called TSP to do different processing depending on which TSP calls it. You can have the calling TSP add information to its call using a SETFIELD control line. The called TSP first tests this field to determine what function to perform.

As shown in this example, you can have a user exit routine control the processing flow of your TSP.

```
SETFIELD   TSCAUFLD=0
LABEL      LOOP
USEREXIT   PROGRAM1               (Sets TSCAUFLD to nonzero
                                   when it completes processing)
TESTFIELD  DONE                   (If TSCAUFLD is not =0, processing
                                   is complete)
⋮

BRANCH     LOOP
LABEL      DONE
RETURN
```

Creating a SETFIELD Control Line

Use the following SETFIELD Specification panel to create a SETFIELD control line:

```
BLM8CU9S               SETFIELD SPECIFICATION               PANEL: ________

Enter 'SETFIELD' control data; cursor placement or input line entry allowed.


                  1. User data.............. 0_______
                  2. Get variable data..... NO_
                  3. Field name............. TSCAUFLD




     When you finish, type END to save or CANCEL to discard any changes.




===>
```

General Rules

If you do not specify user data or variable data on this panel and you specify TSCAUFLD in the **Field name** field, the TSCAUFLD field of the TSCA is reset to blanks (X'40').

If you do not specify user data or variable data on this panel and you specify TSCATLIX in the **Field name** field, the TSCATLIX field of the TSCA is reset to zeros (X'00').

If you specify neither TSCA field name, the default is TSCAUFLD.

Field Descriptions

1. User data

This field indicates the literal data that you want placed in either the TSCAUFLD field or the TSCATLIX field of the TSCA. The data that is entered in this field is collected in the case entered by the user. You can use this data to control the flow within a TSP, or between several TSPs or user exit routines. A user exit routine can access this data.

Valid reply
A mixed string of 1 to 8 characters, or no reply.

Default
No reply.

Restrictions
If you enter data in this field, you must enter **NO** or make no reply in the **Get variable data** field.

2. Get variable data

This field indicates whether you want variable data placed in the TSCA field, TSCAUFLD or TSCATLIX.

Valid reply
YES, NO, or no reply.

Default
NO

Restriction
If you enter **YES** in this field, you must not enter data in the **User data** field. If the SETFIELD control line has **YES** in this field, a user exit routine must move data into the variable data area and set the variable data length or the MOVEVAR control line must move data into the variable data area before processing a SETFIELD control line. The TSCA contains fields for a pointer to the variable data area (TSCAVDAP) and for the length of that data (TSCAVDAL). The pointer field contains the address of a variable data area that is allocated when the TSP environment is initialized. Your exit routine must not modify this pointer.

3. Field name

This field identifies the TSCA field where you want the user data or contents of the variable data area moved.

Valid reply
TSCATLIX, TSCAUFLD, or blank.

Default
TSCAUFLD

Usage Notes

When entering variable data, make sure the variable data length field does not exceed 8 characters when you specify the TSCAUFLD field, or 4 characters when you specify the TSCATLIX field.

For examples, use PMF to look at TSPs BLGTSPPE or BTNTCEAE in your base panel data set.

What the Control Line Does

Tivoli Information Management for z/OS sets the TSCA field TSCAUFLD or TSCATLIX to the specified value. TSCAUFLD is not changed until another SETFIELD control line is processed or the field is modified by a user exit routine. TSCATLIX can be modified by FINDSDATA, another SETFIELD, or a user exit routine.

Another use for the SETFIELD control line is to prevent recursive calls to your TSP. For example, suppose that you modified all your name panels to allow two words. You notice that you have problems searching for some of the names. Further investigation reveals that users who enter **=** into the name fields have their names collected together with one prefix (for example, PERA/JOHN SMITH) while users who enter the actual names have one prefix for each name (for example, PERA/JOHN PERA/SMITH). Rather than restrict the use of the **=**, you can write a TSP to correct the data. This TSP is run after the assisted-entry panel collects the name.

```
FINDSDATA  PERA/. LAST             To find the assignee name just entered
MOVEVAR    TSCASDF                 To move the name
ADDDATA 1, get variable data=yes   To re-enter the name you just entered
PROCESS    EXIT                    To process your responses
LABEL      EXIT
RETURN
```

When your TSP enters the data again, this TSP is processed again. To resolve the problem of this recursive call, you can start the TSP with:

```
TESTFIELD TSCAUFLD = XYZ   if true, branch to exit  (it will never
                            be true the first time)
SETFIELD TSCAUFLD to XYZ   (you would only get here if it was the first
                            time this TSP ran)
```

Return and Reason Codes

After the SETFIELD control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. Table 33 lists those codes.

Table 33. SETFIELD Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0		Successful completion.

Table 33. SETFIELD Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	4	The **Get variable data** field was specified in the control line but the length of the variable data was zero. The function did not end successfully. Your TSP must set up the variable data area and its length before running a SETFIELD control line that specifies variable data. You can do this by calling a user exit routine that sets the variable data length and moves data into it, or by using the MOVEVAR control line. Check your TSP to make sure a USEREXIT or MOVEVAR control line appears in the processing path before the SETFIELD control line that caused the unexpected return code. Add a TRACE control line to your TSP or issue the TRACE command before starting your TSP. If there is a USEREXIT control line in the TSP, check the exit routine's code to make sure it sets the length field properly.
8	8	The user data is longer than the length of the field you want to set. The data was not moved into the field you specified. If you specified field name TSCAUFLD or blanks, only 8 characters can be moved. If you specified field name TSCATLIX, only 4 characters can be moved. If you are using variable data to set these fields, correct your MOVEVAR control line or exit routine's code and make sure that you use the correct amount of data, and set the correct length.
8	12	An internal logic error occurred in National Language Support. Contact your Tivoli representative.
8	16	The user data is not a valid mixed string. Check the data and make the changes required to ensure that you specify valid mixed data.

You can use the TESTFIELD control line to test for these return and reason codes. See "TESTFIELD" on page 175 for information on how to do this.

TSCA Field Usage

Tivoli Information Management for z/OS sets the following TSCA fields after a SETFIELD control line is run. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

TSCAFRET
: Function return code

TSCAFRES
: Function reason code

TSCAUFLD
User field
TSCATLIX
List index value.

If you enter **YES** in the **Get variable data** field and you put data in the variable data field with a user exit routine, the user exit must set the length of the variable data in the following TSCA field:

TSCAVDAL
Current user variable data length.

If you set the variable data with the MOVEVAR control line, the MOVEVAR control line sets field TSCAVDAL for you.

SETRRES

This Remote Data Resource TSX control line is described in "SETRRES" on page 248.

SETTSCA

This control line allows a TSX to set selected fields in the TSCA control block for communications with TSPs and other TSXs.

The SETTSCA Control Line

The format of the SETTSCA control line is:

```
CALL BLGTSX 'SETTSCA','fldname1','fldval1','fldname2','fldval2',...
```

Parameter Descriptions

fldname1

Valid reply
The name of a field to be set. Valid values are

TLIX Used to set TSCATLIX

UFLD Used to set TSCAUFLD

UPTR Used to set TSCAUPTR

VDA Used to set VDA (the Variable Data Area)

fldval

Valid reply
New values for the field. Valid values are dependent on the field being set.

TLIX a number between 0 and 65535

UFLD 0 to 8 characters

UPTR 0 to 4 characters

VDA 0 to 512 characters

Usage Notes and Examples

The SETTSCA control line allows a TSX to set selected fields in the TSCA control block. If the control line is entered with no field names, or a field name is specified with no

associated value, message BLG20011E is issued. If the same field is specified more than once, the last value specified is the one used. The SETTSCA control line does not set the TSCAFRET and TSCAFRES fields.

This is an example of using a SETTSCA control line in a TSX. The field TSCAUFLD is set to the value TEST99.

```
CALL BLGTSX 'SETTSCA','UFLD','TEST99'
```

TESTFIELD

This control line enables you to test any numeric or character field in the TSCA against literal or variable data. Use this control line to test a field in the TSCA to see if it contains the correct data before processing the next control line.

The function provided by the TSP TESTFIELD control line is available in a TSX using conditional program flow controls (if, do, while, etc.) available in the REXX programming language. You can test the values of the TSCA equivalent REXX variables directly. (for example, TSCACPNL). Refer to "TSX Control Lines" on page 54 for more information on the TSCA fields supported by TSX equivalent REXX variables.

This example uses a TESTFIELD control line to test the return code from an exit routine.

```
⋮

USEREXIT   WRITRECS
TESTFIELD  ERROR                 (If return code TSCAFRET is ¬=0,
                                  branch to ERROR)
⋮

LABEL      ERROR
⋮
```

Creating a TESTFIELD Control Line

Use the following TESTFIELD Specification panel to create a TESTFIELD control line:

```
BLM8CU9J               TESTFIELD SPECIFICATION               PANEL: ________

Enter 'TESTFIELD' control data; cursor placement or input line entry allowed.



 1. Field name.........<R> TSCAFRET
    2. Get list index?.... NO_
    3. List index......... 0000
 4. True label.........<R> ERROR___
 5. Get variable data..... NO_
 6. Find string anywhere.. NO_
 7. Find exact length..... NO_
 8. Apply not logic....... YES
 9. Case-sensitive........ NO
10. Test data............. 0_____________________________


     When you finish, type END to save or CANCEL to discard any changes.




===>
```

General Rules

You must specify a LABEL control line within the TSP that identifies the target.

Field Descriptions

1. Field name

This field identifies the field in the TSCA that you want to have tested for the existence of a value. If the result of the test is TRUE, a branch is taken to the control line named in the **True label** field. Otherwise, processing continues with the next control line. This field is required.

Valid reply

A TSCA field name consisting of a string of 1 to 8 SBCS characters.

Default

None.

Restrictions

You must enter the valid name of a TSCA field. See “Terminal Simulator Communications Fields” on page 289 for a list of the field names. You cannot specify a TSCA bit field.

2. Get list index?

This field indicates whether to append the contents of TSCA field TSCATLIX to the contents of TSCARSD. When this field is YES, any value entered in the **List index** field is assumed to be a hexadecimal value and is not treated as a search operator. This field can only be used when the **Field name** field is TSCARSD.

3. List index

This field indicates that the found data was collected with the list processor and that the value in this field should be added to the contents of TSCARSD index in the **Field name** field. The contents of the **Get list index?** field can only be used when the **Field name** field is TSCARSD.

Valid reply

A 1- to 4-byte hexadecimal value.

If you specify **YES** in the **Get list index?** field and specify a value for **List index**, the value in TSCATLIX is added to the value in the **List index** field and the result is appended to the s-word. The resulting sum is treated as a hexadecimal value and not as a search operator.

If you specify **NO** in the **Get list index?** field and specify a value for **List index**, the **List index** field value is appended to the s-word and the value of TSCATLIX is ignored.

4. True label

This field indicates the name of a control line in the current TSP where processing is to resume if the result of the test is TRUE. If the result of the test is FALSE, processing continues with the line following this control line. This field is required.

Valid reply

A mixed string of 1 to 8 characters, beginning with an SBCS alphanumeric or national character or an SO character.

Default

None.

Restrictions

You must enter in this field a valid label name that is identified using a LABEL control line within the current TSP.

5. Get variable data

Valid reply

YES, NO, or no reply.

Enter **YES** when you want the TSCA field identified in the **Field name** field compared to the value pointed to by the TSCAVDAP field. When you enter **NO** or make no reply, this field has no effect.

Default

NO

Restrictions

You cannot use this field if you use the **Test data** field.

If the TESTFIELD control line has **YES** in this field, a user exit routine must move data into the variable data area and set the variable data length or the MOVEVAR control line must move data into the variable data area before processing a TESTFIELD control line. The TSCA contains fields for a pointer to the variable data area (TSCAVDAP) and for the length of that data (TSCAVDAL). The pointer field contains the address of a variable data area that is allocated when the TSP environment is initialized. Your exit routine must not modify this pointer.

6. Find string anywhere

This field is used to specify the method of comparing a character field in the TSCA against a specified value. For either response to this field, the test data must match the data in the TSCA field exactly for the length of the test data. If the response to this field is NO, the test data must begin in the first position of the TSCA field to result in a TRUE condition. If the response is **YES**, the test data may exist anywhere in the TSCA field for a TRUE condition.

Valid reply
YES, NO, or no reply.

Default
NO

Restrictions
You can specify this field as **YES** only if the TSCA field is a character field.

7. Find exact length

This field specifies the type of test to be used on data in a TSCA character string field.

Valid reply
YES, NO, or no reply.

If you enter **YES**, the test result is TRUE only if the test argument exactly matches (in value and length) the field that is being tested.

- If the TSCA field is TSCATBLL, TSCASDF, TSCARPD, TSCARSD, or TSCACMRB, the length of the field data is equal to the value in the associated TSCA length field.
- Otherwise, the length of the field data is considered to be that of the field data with trailing blanks and hexadecimal zeros removed.

If you enter **NO**, the test result is TRUE if the test argument appears anywhere in the field that is being tested.

Default
NO

Restrictions
None.

8. Apply not logic

Valid reply
YES, NO, or no reply.

If you enter **YES**, it indicates that you want the results of the test inverted. In other words, if the result of the test is FALSE, a branch is taken to the **True label** field; if the test is TRUE, processing continues with the next control line.

Default
NO

Restrictions
None.

9. Case-sensitive

Valid reply
YES, NO, or no reply.

A reply of **YES** in this field indicates that you want the character string specified in the **Test data** field to be used as a case-sensitive argument. A reply of **NO** in this field indicates that the case of a character should be ignored when locating data which matches the value specified in the **Literal data** field.

Default

NO

Restrictions

None.

10. Test data

This field indicates the literal data that you want compared against the contents of the TSCA field specified in the **Field name** field.

Valid reply

The valid reply depends on the field type that is being tested:

Numeric field – If you name a TSCA numeric field in the **Field name** field, you can enter values containing the digits 0 though 9. The number is translated to its internal representation (binary) when the control line is processed. A TRUE condition exists when the two values are equal. If you enter **YES** in the **Apply not logic** field, a TRUE condition exists when the two values are not equal.

Character field – If you name a TSCA character field in the **Field name** field, you can enter any string of characters up to the maximum length of the field. The character field named in the **Field name** field is compared against this character string, based on the value in the **Find string anywhere** field. A TRUE condition exists when the test string is found. If you enter YES in the **Apply not logic** field, a TRUE condition exists when the test string is not found.

Default

No reply.

Restrictions

You cannot use this field if you put **YES** in the **Get variable data** field.

When an SBCS comma is required as the first, or only, character of this field, you must precede the SBCS comma with an SBCS space character.

Usage Notes

The TESTFIELD control line provides for conditional branching. You can use the BRANCH control line for unconditional branching.

For examples, use PMF to look at TSPs BTNTAPRV and BTNTA112 in your base panel data set.

If you want to use literal data for the test, enter the data in the **Test data** field. If you want to use variable data, you must design your TSP to put data into the variable data area before processing the TESTFIELD control line. This can be done with either a USEREXIT or a MOVEVAR control line.

You cannot include the s-word watermark characters X'BA', X'BB', X'BC', X'BD', X'BE', or X'BF' as literal or variable data when building a search argument. Therefore, when using the TESTFIELD control line to test for a returned s-word, the input field should contain the

s-word without the watermark character and you should enter YES in the **Find string anywhere** field. You can use **Get list index?** and **List index** fields to allow you to test for specific s-words added by the list processor program exit.

All but 2 character fields in the TSCA are set to blanks (X'40') when the TSP environment is initialized. TSCARRB, TSCACMRB, and TSCAVDAL are set to binary zeros. If you want to check whether a field has been used, specify the appropriate characters (blanks or binary zeros) in the variable data area as the comparison data. Any blanks specified in the literal data area are stripped out when the control line processes.

When you use TESTFIELD to test the results from a FINDSDATA control line that looked for any occurrence of a prefix (for example, look for an assignee name in the record by specifying PERA/.), note that the return code is zero, even if the user has blanked out this field. Therefore, a more accurate test would be against TSCASDFL, the length of the returned data, rather than against TSCAFRET.

What the Control Line Does

If the TESTFIELD control line results in a true condition, a branch is taken to the target you specified in the control line. If not logic is used, the branch is taken when the result of the test is false.

Return and Reason Codes

Running a TESTFIELD control line does not change the setting of the TSCA return or reason codes.

TSCA Field Usage

Tivoli Information Management for z/OS sets the following TSCA fields after a TESTFIELD control line is run. For more information about these fields, see “Terminal Simulator Communications Fields” on page 289.

TSCACPOS
: Byte number where string is found

If you enter **YES** in the **Get variable data** field and you put data in the variable data area with a user exit routine, the user exit must set the length of the variable data in the following TSCA field:

TSCAVDAL
: Current user variable data length

If you set the variable data with the MOVEVAR control line, the MOVEVAR control line sets field TSCAVDAL for you.

TESTFLOW

This control line enables you to test for the panel or message ID identified within the control line.

When you are ready to process an IRC using a TSP, be sure that it is entered from the correct panel. You can use TESTFLOW to verify that the current panel name in the TSCA is correct before you process any data for that panel. As shown in the following example, you can also use TESTFLOW as the first line of your TSP to test for the panel that called the TSP.

The function provided by the TSP TESTFLOW control line is available in a TSX using conditional program flow controls available in the REXX programming language. You can test the values of the TSCA equivalent REXX variables directly (for example, TSCACPNL). Refer to "TSX Control Lines" on page 54 for more information on the TSCA fields supported by TSX equivalent REXX variables. You can also test for messages by examining compound variable BLG_MESSAGE. Refer to "Testing Terminal Simulator Panels (TSPs) and EXECS (TSXs)" on page 251 for more information on TSP and TSX debugging techniques.

In this example, the TESTFLOW control line at the beginning of the TSP tests whether the current panel is the System application Primary Options Menu.

```
TESTFLOW   SYSTEM                 (Test for BLG0EN10, System application
                                    If yes, branch to label SYSTEM)
ADDDATA    6,1,SE + PERA/SMITH     (Otherwise, you are in
                                    Management.)
BRANCH     NEXT
LABEL      SYSTEM
ADDDATA    3,2,6,1,SE + PERA/SMITH  (3,2 puts you at the
                                    management panel)
LABEL      NEXT
⋮
```

You can also use this control line to test which path an IRC used. For example, if you delete a record, one of two panels will appear, depending on whether the record was found. You can use TESTFLOW to check which path you took. If your TSP creates a record, you might want to use TESTFLOW to check for the *Record filed successfully* message.

Creating a TESTFLOW Control Line

Use the following TESTFLOW Specification panel to create a TESTFLOW control line:

```
BLM8CU9K                TESTFLOW SPECIFICATION              PANEL: ________

Enter 'TESTFLOW' control data; cursor placement or input line entry allowed.


                   1. Verify name........... BLG0EN10
                   2. Verify type........<R> PANEL__
                   3. True label.........<R> SYSTEM__
                   4. Get variable data..... NO_
                   5. Apply not logic....... NO_




     When you finish, type END to save or CANCEL to discard any changes.









===>
```

General Rules

You must specify a LABEL control line within the TSP that identifies the target of the **True label** field.

Field Descriptions

1. Verify name

Valid reply

An 8-character name of a panel created through PMF or one of the base panels shipped with Tivoli Information Management for z/OS, a Tivoli Information Management for z/OS message ID, or no reply.

This is the name you want to test for when this control line is processed. If the result of the test is TRUE, a branch is taken to the control line named in the **True label** field. Otherwise, processing continues with the next control line.

Default

No reply.

Restrictions

You cannot use this field if you enter **YES** in the **Get variable data** field.

The panel name must consist of SBCS characters only.

2. Verify type

This field indicates whether the test value is a panel name or a message ID. If you specify PANEL, a true condition exists when the currently displayed panel matches the test value. If you enter MESSAGE, a true condition exists when the test value is a message ID on the current message chain, a false condition exists when the test value is a message ID on the saved message chain or the message ID does not exist on either message chain. This field is required.

Valid reply

PANEL or MESSAGE

Default

PANEL

Restrictions

None.

3. True label

This field contains the name of a control line in the current TSP where processing resumes if the result of the test is true. If the result of the test is false, processing continues with the line following this control line. This field is required.

Valid reply

A mixed string of 1 to 8 characters, beginning with an SBCS alphanumeric or national character or an SO character.

Default

None.

Restrictions

You must enter a valid label name that is identified by a LABEL control line within the current TSP.

4. Get variable data

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that the panel name or message ID you want to test for should be taken from the variable data area. When you enter **NO** or make no reply, this field has no effect.

Default

NO

Restrictions

You cannot use this field if you enter data in the **Verify name** field.

If the TESTFLOW control line has YES in this field, a user exit routine must move data into the variable data area and set the variable data length or the MOVEVAR control line must move data into the variable data area before processing the TESTFLOW control line. The TSCA contains fields for a pointer to the variable data area (TSCAVDAP) and for the length of that data (TSCAVDAL). The pointer field contains the address of a variable data area that is allocated when the TSP environment is initialized. Your exit routine must not modify this pointer.

5. Apply not logic

Valid reply

YES, NO, or no reply.

If you enter **YES** in this field, it indicates that you want the results of the test inverted. In other words, if the result of the test is TRUE, it is treated as FALSE; and if the result is FALSE, it is treated as TRUE.

Default

NO

Restrictions

None.

Usage Notes

The TESTFLOW control line provides for conditional branching. You can use the BRANCH control line for unconditional branching.

This control line sets return and reason codes based upon its results. Because of this, it recommended that you check the results of a PROCESS control line before using TESTFLOW.

When testing for help panels, be aware that the current panel name (TSCACPNL) for help panels is always set to BLG1T007.

For examples, use PMF to look at TSPs BTNTAM01 and BTNTARUP in your base panel data set.

What the Control Line Does

If you enter a message ID in the **Verify name** field, it is compared to every message ID on the current message chain to see if a match exists. If there is a match, a true condition is set. If the message ID exists on the saved message chain or does not exist on either message

chain, then the test is false. If the message ID exists on the saved message chain, then TSCAFRET is set to 4 and TSCAFRES is also set to 4.

If you enter a panel name in the VERIFY NAME field, it is compared to the current Tivoli Information Management for z/OS panel. If the panel name matches the current panel, a TRUE condition is set; if not, the test returns FALSE.

If the TESTFLOW control line results in a TRUE condition, a branch is taken to the target. You must specify a target in a LABEL control line within the same TSP. If not logic is used, the branch is taken when the result of the test is FALSE.

Return and Reason Codes

After a TESTFLOW control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 34.

Table 34. TESTFLOW Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0		Successful completion.
0	4	The requested operation was successful, but an abnormal condition occurred. The variable data is not valid for a message or panel name. Check your user exit routine's code to make sure the length of the data is no more than 8 characters and that the message ID or panel name starts with an alphabetic character. You can have your TSP print the TSCA to see what your user exit routine is actually entering.
4	4	The message was found, but it is on the saved chain rather than on the current chain. Either the message could have been on the saved message chain when the TSP began, or it could have been moved from the current chain to the saved chain by a PROCESS or MESSAGE control line.

TSCA Field Usage

Tivoli Information Management for z/OS sets the following TSCA fields after a TESTFLOW control line is run. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

TSCAFRET
: Function return code

TSCAFRES
: Function reason code.

If you enter **YES** in the **Get variable data** field and you set the variable data with a user exit routine, the user exit routine must set the length of the variable data in the following TSCA field:

TSCAVDAL
: Current user variable data length

If you set the variable data with the MOVEVAR control line, the MOVEVAR control line sets the field TSCAVDAL for you.

TRACE

This control line traces the flow of the control lines being processed by one or more TSPs. It can also be used to initiate REXX tracing in a TSX if Trace LINK function is set to **YES** on the TSP TRACE control line and the TSX is coded to support tracing (the TSX checks the value of variable BLGTRACE). You can turn TRACE on or off at any point in a TSP. It can trace some or all of the control lines processed in a single TSP, and can include control lines of any other TSP that is processed with the LINK control line or the 002B function code. (See "The 002B Function Code" on page 259 for more information on starting a TSP using the 002B function code.)

Refer to "TSX Considerations" on page 187 for information about using the TRACE control line in a TSX.

You can use TRACE to check the flow of your TSP. Say, for instance, the length field for the variable data area was not set properly before an ADDDATA control line was processed. A TRACE report shows if a USEREXIT or MOVEVAR control line, which sets this field, was processed before the ADDDATA.

This example uses a TRACE control line to produce a listing of certain control lines in a TSP.

```
LABEL      TSP00001              LABEL   TSP00002
TRACE      TRACE=YES             .
           LINK=YES              .
LINK       TSP00002              RETURN
⋮

LINK       TSP00003              LABEL   TSP00003
.                                .
.                                .
.                                RETURN
TRACE      TRACE=NO
LINK       TSP00004
⋮
```

This example produces a listing of the control lines that were run. Suppose that TSP00004 does not contain a TRACE control line. This listing would include all control lines run by TSP00002 and TSP00003, but only the control lines run by TSP00001 up to the LINK TSP00004 control line.

Creating a TRACE Control Line

Use the following TRACE Specification panel to create a TRACE control line:

```
BLM8CU9N                TRACE SPECIFICATION                  PANEL: TSP00001

Enter 'TRACE' control data; cursor placement or input line entry allowed.



                    1. Set TRACE on.......... YES
                    2. Trace LINK function... YES







      When you finish, type END to save or CANCEL to discard any changes.









===>
```

Field Descriptions

1. Set TRACE on

This field sets control line tracing on or off. When control line tracing is on, all control lines that are processed in the current TSP are formatted and written to a specified output destination. When tracing is off, control lines are not written to the output destination. You can use any number of TRACE control lines to turn tracing on or off.

Valid reply

YES, NO, or no reply.

Default

YES

Restrictions

When a TSP is exited (either by processing the last control line or by processing a RETURN control line), tracing returns to its former setting.

2. Trace LINK function

Valid reply

YES, NO, or no reply.

If you enter **YES** in this field, control lines in all of the subsequent linked-to TSPs are traced until tracing is set off or you enter **NO** in this field. If you enter **NO**, only control lines in the current TSP are traced.

Default

NO

Restrictions

If you enter **YES** in this field, you must also enter **YES** in the **Set TRACE on** field. When a TSP ends (either by running the last control line or a RETURN control line), tracing returns to its former setting.

Usage Notes

See "Using the TSP TRACE Control Line" on page 254 for information on how to use TRACE to help you analyze a TSP's flow.

The TRACE command provides the same capability as the TRACE control line, except that you specify the trace on the command line at the time you run the TSP, not *in* the TSP. Therefore, you need not update your TSP to remove the TRACE control line when you are ready to go to production. For more information on the TRACE command, refer to the *Tivoli Information Management for z/OS User's Guide*.

What the Control Line Does

TRACE creates a report that follows the processing path of a TSP. The report consists of a description of each control line in the TSP. An example of this report is shown in Figure 6 on page 255.

The output from the TSP TRACE control line is sent to the destination LRECL parameter.you defined on the BLGTRACE DD statement, to the SYSOUT device or to another output data set. You must allocate the BLGTRACE DD statement before calling the TSP. If you direct the output to another data set, the DCB information for the output data set must include RECFM=VBA and LRECL=137. See the *TSO/E Command Reference* manual for additional information on allocation.

Note: TSX output goes to the destination determined by REXX.

Note: Running traces on multiple sessions concurrently can produce unexpected results. This applies to the TRACE command as well as the TRACE control line.

The completeness of the trace depends upon the output destination specified in the BLGTRACE DD statement. If the output destination is a data set, trace information for some sessions might not be recorded. If the output destination is a SYSOUT device, trace information for all sessions is recorded completely.

TSX Considerations

The variable BLGTRACE will be set to 1 when a TSX is started if the TRACE command was used to turn tracing on or if the TSX was started via the TSP LINK control line with Trace LINK Function set to **YES**. It is the responsibility of your TSX to check the value of BLGTRACE and specify the desired REXX TRACE parameters you prefer. Example REXX code:

```
IF BLGTRACE=1 THEN, /* TSP/TSX Tracing Active?              */
  TRACE RESULTS     /* Start REXX trace with desired options */
```

Return and Reason Codes

Running a TRACE control line does not change the setting of the TSCA return or reason codes.

TSCA Field Usage

The TRACE control line does not use any TSCA fields.

UNFLATTEN

This control line takes a record that was previously extracted from a database by the FLATTEN control line, and stores the record in a database that is equivalent to the one from which it was originally obtained.

CAUTION:
You can damage your existing database if you do not use this control line correctly. For information on the security measures you can use to protect against its misuse, see the discussion of data integrity and security using TSPs and TSXs in the *Tivoli Information Management for z/OS Planning and Installation Guide and Reference***.**

Refer to "The UNFLATTEN TSX Control Line" on page 190 for information about using the UNFLATTEN control line in a TSX.

This example uses an UNFLATTEN control line in a TSP that returns records to a system application database.

```
LABEL      RESREC
USEREXIT   OPENSEQF              (Opens sequential data set and
                                  obtains buffer)
LABEL      NEXT
USEREXIT   READFLAT              (Reads flattened record)
TESTFIELD  TSCAFRET = 4          (If true, branch to label ALLDONE)
UNFLATTEN
TESTFIELD  TSCAFRET ¬=0           (If true, branch to label ERROR)
BRANCH     NEXT
LABEL      ALLDONE
USEREXIT   CLOSSEQF              (Closes sequential data set and
                                  releases buffer)
RETURN
LABEL      ERROR
PRINT                            (Print messages, panel, and TSCA fields)
BRANCH     ALLDONE
```

Creating an UNFLATTEN Control Line

Use the following UNFLATTEN Specification panel to create an UNFLATTEN control line:

```
BLM8CU9X            UNFLATTEN  SPECIFICATION                PANEL: ________

Enter 'UNFLATTEN' control data; cursor placement or input line entry allowed.


                1. Retain record id.......<R> YES




       When you finish, type END to save or CANCEL to discard any changes.









===>
```

General Rules

Because of the USEREXIT considerations listed on 194, use assembler language to write user exit routines to perform the file operations associated with UNFLATTEN.

Field Descriptions

1. Retain record ID

Valid reply

YES or **NO**

A reply of **YES** indicates that the system-assigned ID that exists in the record is to be used for the unflattened record, or that the record has a user-assigned ID. A reply of **NO** indicates that a new record ID is to be assigned by the system when the record is unflattened. This field is required.

Default

YES

Restrictions

This field is ignored if the record being unflattened has a user-assigned record ID, which is always retained.

Usage Notes

You are responsible for acquiring storage and releasing storage to use as a buffer for UNFLATTEN. You are also responsible for sequentially moving records from where they were moved during FLATTEN processing to the UNFLATTEN buffer. Before processing an UNFLATTEN control line, you must set the TSCAUFBL field to the length of this buffer. Additionally, you must set the TSCAUFBP field to the storage address where the flattened record resides. This is done with a USEREXIT control line calling a user-written exit routine.

To maintain the relationships between records, you should retain record IDs when records are unflattened. It is your responsibility to make sure that all related records are unflattened. Failure to maintain the relationships between records can damage the database. The database into which records are unflattened should only be used for that purpose. You must not mix records from different databases.

The UNFLATTEN control line updates the variable data area and the variable data area length. If later in your TSP you are going to use the information that is in the variable data area, save the information before the UNFLATTEN control line is processed and restore it before it is needed. If you only need to see the information, print it using a PRINT control line before UNFLATTEN is processed.

Logically Partitioned Database Considerations

If you are using logically partitioned databases (described in the *Tivoli Information Management for z/OS Program Administration Guide and Reference*), caution must be exercised when unflattening records with system-assigned (numeric) record ids (RNIDs). The UNFLATTEN control line does not modify the last entry number for the partition, so it is possible to unflatten a record containing an RNID that is higher than the last entry number. If new records are subsequently created, they may have the same RNID (obtained from the last entry number value) as one of the records that was unflattened.

If the database into which records are unflattened is logically partitioned and the active privilege class has Universal Partition Access authority, the records that are unflattend retain the Owning Partition Name of the partition from which they were flattened. Appropriate checks are made to avoid duplicate RNIDs in the target logical partition.

If the database into which records are unflattened is logically partitioned and the active privilege class does **not** have Universal Partition Access authority, the Primary Partition Name from the privilege class of the user performing the UNFLATTEN becomes the Owning Partition Name of the unflattened records. If that user's privilege class does not contain a Primary Partition Name, the unflattened records will not be assigned an Owning Partition Name.

What the Control Line Does

When UNFLATTEN runs successfully, the ID of the record is stored in the variable data area pointed to by the TSCA, and the variable data length field in the TSCA is set to the length of the record ID. If you entered **NO** in the **Retain record id** field, this may not be the same as the record ID of the record prior to unflattening. User-assigned record IDs are always retained.

The UNFLATTEN TSX Control Line

The format of the UNFLATTEN control line is:

```
CALL BLGTSX 'UNFLATTEN',segcnt,segsize,stemname,options
```

Parameter Descriptions

1. segcnt

Valid reply

The number of segments (elements) the flattened record is broken into.

Default

None

Required

2. segsize

Valid reply
The size of each segment.

Default
None

Required

3. stemname

Valid reply
The stem of a REXX compound variable which contains the flattened record.

Default
BLG_FLATTEN.

Optional

4. options

Valid reply
Options associated with the record to be unflattened. You can specify multiple options in any order, separating each parameter with a comma.

ORIGINAL|ASSIGN
ORIGINAL indicates that the unflattened record should have the same system-assigned RNID as the flattened record; ASSIGN indicates that the unflattened record should be assigned a new system-assigned RNID.

NOHISTORY
Indicates that the history data for this record is not copied when the flattened record is written to the buffer. The default (no value) is to copy the history data with the record.

NOTEXT
Indicates that the freeform text data for this record is not copied when the flattened record is written to the buffer. The default (no value) is to copy the freeform text with the record.

REPLACE
Indicates that the unflattened record should replace an existing record in the database when the unflattened record has the same RNID as an existing record. This keyword is ignored if ASSIGN is specified for the system-assigned RNIDs.

no value specified
See Default.

Default
ORIGINAL is the default for the ORIGINAL|ASSIGN option; the default for the other parameters is that history data is copied with the record and freeform text is copied with the record.

Optional

Usage Notes and Examples

This example uses an UNFLATTEN control line in a TSX. Also refer to the BLGUNFLT data set member of the SBLMTSX data set.

```
member='R0003168'

'ALLOC FI(FLATPDS) DA('BLM.FLATPDS('member')'') SHR'
'EXECIO * DISKR FLATPDS (FINIS STEM FLAT.'

if rc=0 then do
  /* Unflatten the record.  Assign a new record ID if necessary */
  CALL BLGTSX 'UNFLATTEN',FLAT.0,LENGTH(FLAT.1),'FLAT.','ASSIGN'

  if tscafret=0 then           /* FLATTEN successful?           */
    msgtext='Unflattened record' tscavda 'created successfully'
  else
    msgtext='Unflatten failed.  Return code=('tscafret','tscafres')'
  CALL BLGTSX 'MESSAGE',,msgtext,'SAVE'
  end

'FREE FI(FLATPDS)'
```

Return and Reason Codes

After an UNFLATTEN control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 35.

Table 35. UNFLATTEN Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0		Successful completion.
4	4	Successful completion, but the variable data area was not empty when the record ID was stored.
8	4	The UNFLATTEN buffer pointer in the TSCA was zero. The record was not restored. You must run a user exit routine to get a buffer, set the pointer field (TSCAUFBP) in the TSCA to point to the address of the unflattened record, and set the length of the unflattened record (TSCAUFBL).
8	8	The UNFLATTEN buffer was not in the expected format. The record was not restored. Verify that your user exit routine moved a flattened record into the unflatten buffer prior to processing the UNFLATTEN control line. If the flattened record was modified, it might not be in the proper format, so it cannot be unflattened.
8	12	The record ID already exists in the target database. The ID of the current record is returned in the variable data area. Variable data that already exists in the variable data area was lost. The record was not restored. If the record ID is a system-assigned ID, consider retrying the unflatten process, but do not retain the record ID. If the record ID is a user-assigned ID, consider copying the existing record, then change the ID, and retry the unflatten process.

Table 35. UNFLATTEN Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	16	There is not enough storage available to complete the function. The record was not restored. Increase the region size and try again.
8	20	Tivoli Information Management for z/OS could not file the record. Make sure that your database is not damaged. If it is, try again. Check with your system programmer; you might have an I/O error. If the problem continues, notify your Tivoli representative.
8	24	An internal logic error occurred in National Language Support. Contact your Tivoli representative.
8	28	The record contains mixed data that is not valid. Check the data and make the changes required to ensure that the record contains only valid mixed data.
8	32	The record ID could not be REPLACEd in the database. You must have DELETE authority to use the REPLACE option for this record. If you do have DELETE authority for this record, then check with your system programmer; you might have an I/O error.

You can use the TESTFIELD control line to test for these return and reason codes. See "TESTFIELD" on page 175 for information on how to do this.

TSCA Field Usage

Tivoli Information Management for z/OS sets the following TSCA fields after an UNFLATTEN control line is run. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

TSCAFRET
: Function return code

TSCAFRES
: Function reason code

TSCAVDAL
: Current user variable data length.

A user exit routine must set the following TSCA fields:

TSCAUFBP
: Pointer to unflattened record

TSCAUFBL
: Size of the unflatten buffer.

USEREXIT

A USEREXIT control line provides a communication link between Tivoli Information Management for z/OS and a user exit routine.

Refer to "The USEREXIT TSX Control Line" on page 207 for information about using the USEREXIT control line in a TSX.

It allows you to start a user exit routine that retrieves data stored in the TSCA. The TSCA is the single parameter passed to the user exit routine. In this way, a routine has access to all the data collected by the control line.

Before writing a user exit routine, you must be familiar with the contents of the TSCA. The TSCA is divided into two sections. The first section contains fields that are set when a control line is run, such as return and reason codes, data found by FINDSDATA, and pointer fields. The fields in the second section contain the data you entered when creating the USEREXIT control line, such as s-word information, bit flag data (flag fields), and TSCA field names. You can use as many fields in this section as necessary for input to your user exit routine. Your user exit routine can then interpret these fields in any way.

You can change the data in the TSCA fields. However, modifying some fields (the variable data area pointer field, for example) causes problems. The table in "Terminal Simulator Communications Fields" on page 289 describes each field in the TSCA and shows which fields you must not modify. This appendix also contains an assembler language mapping of the TSCA, which you need for your user exit routine.

Note: User exit routines can be run in storage above 24-bit addressing.

You can write a TSP exit routine using assembler, PL/I or VS COBOL II. Consider the following when writing a user exit routine.

- PL/I and VS COBOL II initialize the processing environment each time the user exit routine is called and end the environment each time control returns to the TSP. This overhead may significantly degrade performance if the user exit routine is called many times, as it would be within a loop.
- User-exit routines written in PL/I and VS COBOL II initiate and end any kind of file operation (that is, the file is opened, processed, and closed each time the user exit is used).
- When you write a user exit routine in assembler language, you can leave a file, and it remains open when control is returned to the TSP. You can retain the DCB address in the TSCA and incorporate logic in the user exit routine to determine when to close the file. For this reason, if you plan to use a user exit routine that repeatedly accesses a file by multiple invocations of a USEREXIT control line, it is recommended that you use assembler language.

If you write your user exit routine in PL/I, you need to be aware of the following:

- You must compile the subprogram with the MAIN option.
- You must call the subprogram using entry point PLICALLA (specify ENTRY PLICALLA when link editing).
- If the subprogram is receiving a control block as a parameter, use the following sample code to avoid the possibility of PL/I locators or descriptors adding another level of indirection and an 0C4 ABEND code.

```
TSP: PROC(PARM) OPTIONS(MAIN);
DCL PARM FIXED,                  (length of TSCA)
    PTR POINTER,
    1 TSCA BASED(PTR),
     2 TSCAACRN CHAR(4),
     .
     .
     .
```

```
     ........;
/* End of declarations. Begin executable code */
PTR=ADDR(PARM);
/* TSCA is now usable at this point */
⋮

END TSP;
```

- PL/I processes numeric data in packed decimal format. Declare control block areas that receive numeric data as fixed binary so that a true binary value is received, not packed decimal data.

USEREXIT Linkage Conventions

Tivoli Information Management for z/OS uses standard linkage conventions and sets up a parameter list for calling a user-written exit routine. The contents of the general purpose registers upon entry to a user exit routine follow:

Register	**Content**
0	Unpredictable
1	Address of a 1-word input parameter - TSCA
2-12	Unpredictable
13	Address of a 72-byte register save area
14	Return address
15	Module entry-point address

This is the parameter list (PLIST) as it appears to an assembler language routine.

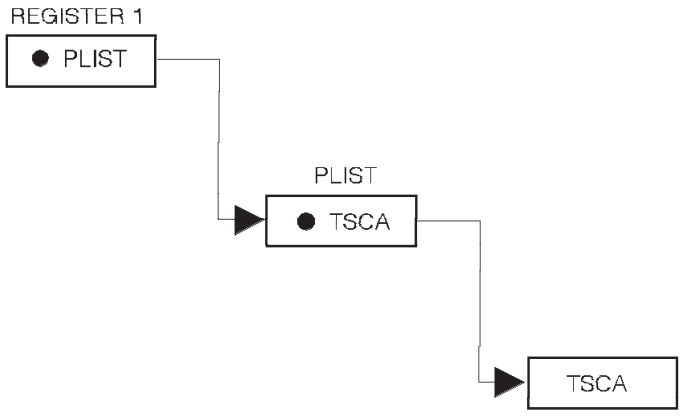


Figure 5. USEREXIT Linkage Structure

Creating a USEREXIT Control Line

Use the following Control Line Summary panel to create a USEREXIT control line:

```
BLM8CU9L                  CONTROL LINE SUMMARY                PANEL: ________



 Label name.............. ________        S-word index..... ____
 TSCA field name......... ________        Structured word.......... __________
 Apply not logic......... ___             Prefix word index........ ____
 Get variable data....... ___             P-word................... ______
 Panel name.............. ________        Validation............... __________
 Find string anywhere.... ___             Function exit............ ________


  Select one of the choices, or type END to save or CANCEL to discard changes.


                      1. Data field specification.
                      2. Flag field specification.




===>
```

The fields at the top of the panel represent a summary of information contained in the control line and are display-only fields. This information is added to the panel after you create the control line. For more information on *Data field* specification, see "Specifying Input Data"; for more information on *Flag field* specification, see "Setting Internal Flag Fields" on page 201.

Specifying Input Data

The following panel is displayed when you select option 1, **Data field specification**, from the Control Line Summary panel:

```
BLM8CU9P               DATA FIELD SPECIFICATION               PANEL: ________

Enter 'USEREXIT' data fields; cursor placement or input line entry allowed.

 1. Function exit...........<R> ________      Structured word...... __________
 2. Structured word index...... ____          Word acronym......... __________
 3. Prefix index............... ____          Prefix............... ______
 4. Label name................. ________      Validation........... __________
 5. Panel name................. ________      New structured word.. __________
 6. Verify name................ ________      New word acronym..... __________
 7. TSCA field name............ ________      New prefix........... ______
 8. New structured word index.. ____          New validation....... __________
 9. New prefix index........... ____
10. User data.................. ________
11. List index................. _____
12. Literal/Test data.......... ____________________________________

13. New data................... ____________________________________

      When you finish, type END to save or CANCEL to discard any changes.


===>
```

General Rules for Panel BLM8CU9P

The following definitions describe the fields as they are normally used in a TSP. As mentioned earlier, you can give other meanings to these fields for this control line.

The following field descriptions name the TSCA field where the data is stored. This is provided so you can associate the fields on the specification panel to your user exit routines. The only field that is required is the **Function exit** field, which is the name of your user exit routine.

These fields are set only during processing of the called user exit routine. The next control line in your TSP modifies these fields.

Field Descriptions for Panel BLM8CU9P

1. Function exit

This field indicates the name of the user exit routine that receives control when this control line is run. This field is required.

Valid reply
The name of a user exit routine that is 1 to 8 alphanumeric SBCS characters long.

TSCA field
TSCAFUEX

Default
None.

Restrictions
The function exit name must refer to an executable load module.

2. Structured word index

Valid reply
The index of an s-word in the dictionary data set or no reply.

TSCA fields
TSCASIX, TSCASWDL, TSCASWD

Default
No reply.

Restrictions
None.

3. Prefix index

Valid reply
The index key of a p-word in the dictionary data set, or no reply.

TSCA fields
TSCAPIX, TSCAPFXL, TSCAPFX

Default
No reply.

Restrictions
None.

4. Label name

Valid reply
A mixed string of 1 to 8 characters, beginning with an SBCS alphanumeric or national character or an SO character.

TSCA fields
TSCALABL, TSCATNUM

Default
No reply.

Restrictions
The name is not validated as a true TSP label name.

5. Panel name

Valid reply
An 8-character panel name or no reply.

TSCA field
TSCAPANL

Default
No reply.

Restrictions
The panel name must consist of SBCS characters only.

6. Verify name

Valid reply
A string of 8 characters or no reply.

TSCA field
TSCAVNAM

Default
No reply.

Restrictions
None.

7. TSCA field name

Valid reply
A valid TSCA field name or no reply.

TSCA field
TSCAFLD

Default
No reply.

Restrictions
None.

8. New structured word index

Valid reply
The index of a new s-word in the dictionary data set, or no reply.

TSCA fields
TSCANSIX, TSCANSDL, TSCANSWD

Default
No reply.

Restrictions
None.

9. New prefix index

Valid reply
The index key of a new p-word that exists in the dictionary data set, or no reply.

TSCA fields
TSCANPIX, TSCANPFL, TSCANPFX

Default
No reply.

Restrictions
None.

10. User data

This field indicates to the SETFIELD control line the static data that is to be used.

Valid reply
A string of 1 to 8 characters or no reply.

TSCA fields
TSCAIFLD, TSCAIFLL

Default
No reply.

Restrictions
None.

11. List index

Valid reply
A 1- to 4-byte hexadecimal value.

12. Literal/Test data

Valid reply
A string of 1 to 32 characters, or no reply.

TSCA fields
TSCALIT, TSCALITL

Default
No reply.

Restrictions
When an SBCS comma is required as the first, or only, character of the **Literal data** field, you *must* precede the SBCS comma with an SBCS space character.

13. New data

Valid reply
A string of 1 to 32 characters, or no reply.

TSCA fields
TSCANDAT, TSCANDAL

Default
No reply.

Restrictions
None.

Structured word

If you enter an s-word index when you create the control line, this field is filled in automatically. It displays the actual s-word found in the dictionary.

Word acronym

If you enter an s-word index when you create the control line, this field is filled in automatically. It is part of the s-word entry in the dictionary. This field could be blank if a word acronym does not exist for the entry.

Prefix

If you enter a prefix index when you create the control line, this field is filled in automatically. It displays the actual prefix found in the dictionary.

Validation

If you enter a p-word index when you create the control line, this field is filled in automatically. It displays the validation pattern associated with the prefix index.

New structured word

If you enter a new s-word index when you create the control line, this field is filled in automatically.

New word acronym

If you enter a new s-word index when you create the control line, this field is filled in automatically. It is part of the s-word entry in the dictionary. This field could be blank if a word acronym does not exist for the entry.

New prefix

If you enter a new prefix index when you create the control line, this field is filled in automatically. It displays the actual prefix obtained from the dictionary.

New validation

If you enter a new prefix index when you create the control line, this field is filled in automatically. It displays the validation pattern associated with the p-word index.

Setting Internal Flag Fields

Flag fields are available as input to your user exit routines. By themselves, these flags do not cause any actions to occur; they are merely bits that can be set to indicate processing conditions. For example, setting the **Print the messages** flag bit to ON does not cause messages to be printed, but your user exit routine can detect that the flag bit is on, and it can use that information according to your design.

The following panel is displayed when you select option 2, **Flag field specification**, from the Control Line Summary panel.

```
BLM8CU9Q              FLAG FIELD SPECIFICATION              PANEL: ________

Enter 'USEREXIT' flag fields; cursor placement or input line entry allowed.

 1. Verify type.............. _______    12. Use id of current record..... ___
 2. Word occurrence......... _____      13. Use id of last record filed.. ___
 3. Apply not logic......... ___        14. Retain record id............. ___
 4. Get variable data....... ___        15. Save generated message....... ___
 5. Treat as string data.... ___        16. Insert data type............. ____
 6. Find string anywhere.... ___        17. Replace data?................ ___
 7. Set TRACE on............ ___        18. Get list index?.............. ___
 8. Trace LINK function..... ___
 9. Print the messages...... ___
10. Print the screen........ ___
11. Print the TSCA.......... ___



      When you finish, type END to save or CANCEL to discard any changes.



===>
```

General Rules for Panel BLM8CU9Q

The following definitions describe the fields as they are normally used in a TSP. You can give other meanings to these fields for this control line. The only requirement is that the data you enter must meet the validation criteria.

The following field descriptions name the TSCA field where the data is stored. This is provided so you can associate the fields on the specification panel to your user exit routines.

These fields are set only during processing of the called user exit routine. The next control line in your TSP resets these fields.

Field Descriptions for Panel BLM8CU9Q

1. Verify type

Valid reply

PANEL, MESSAGE or no reply.

Your reply to this field results in a bit in the control line being set on or off. If the reply is MESSAGE, the bit is set on; otherwise, the bit is set off.

TSCA field

TSCA0FLO

Default
No reply.

Restrictions
None.

2. Word occurrence

Valid reply
FIRST, NEXT, LAST, PREV, or no reply.

The valid replies to this field are FIRST, NEXT, LAST, and PREV. Three bits in the control line indicate your response and determine the setting of this command.

If you enter FIRST, one bit is on; if you enter LAST, another bit is on; and if you enter NEXT or make no reply, both bits are off. If you enter PREV, the third bit is on; otherwise for NEXT, all three bits are off. A control panel, which is the target of the **Word occurrence** field assisted-entry panel, adds the necessary structured description entries to ensure the proper bits are set on or off.

TSCA fields
TSCA0FST, TSCA0LST, TSCA2PRV

Default
No reply.

Restrictions
None.

3. Apply not logic

Valid reply
YES, NO, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

TSCA field
TSCA1ANL

Default
No reply.

Restrictions
None.

4. Get variable data

Valid reply
YES, NO, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

TSCA field
TSCA0VAR

Default
No reply.

Restrictions
None.

5. Treat as string data

Valid reply
YES, NO, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

TSCA field
TSCA1STG

Default
No reply.

Restrictions
None.

6. Find string anywhere

Valid reply
YES, NO, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

TSCA field
TSCA0FND

Default
No reply.

Restrictions
None.

7. Set TRACE on

Valid reply
YES, NO, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

TSCA field
TSCA0TRO

Default
No reply.

Restrictions
None.

8. Trace LINK function

Valid reply
YES, NO, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

TSCA field
TSCA0TRL

Default
No reply.

Restrictions
None.

9. Print the messages

Valid reply
YES, NO, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

TSCA field
TSCA1MSG

Default
No reply.

Restrictions
None.

10. Print the screen

Valid reply
YES, NO, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

TSCA field
TSCA1PNL

Default
No reply.

Restrictions
None.

11. Print the TSCA

Valid reply
YES, NO, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

TSCA field
TSCA1TSC

Default
No reply.

Restrictions
None.

12. Use ID of current record

Valid reply

YES, NO, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

TSCA field

TSCA1CRD

Default

No reply.

Restrictions

None.

13. Use ID of last record filed

Valid reply

YES, NO, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

TSCA field

TSCA1LRD

Default

No reply.

Restrictions

None.

14. Retain record ID

Valid reply

YES, NO, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

TSCA field

TSCA1RET

Default

No reply.

Restrictions

None.

15. Save generated message

Valid reply

YES, NO, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

TSCA field

TSCA2SGM

Default

No reply.

Restrictions
None.

16. Insert data type

Valid reply
HEX, CHAR, or no reply.

Your reply to this field results in a bit in the control line being set on or off. If you enter HEX, the bit is set on. If you enter CHAR or make no reply, the bit is set off.

TSCA field
TSCA2ITD

Default
No reply.

Restrictions
None.

17. Replace data?

Valid reply
YES, NO, or no reply

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

Default
NO

18. Get list index?

Valid reply
YES, NO, or no reply

Your reply to this field results in a bit in the control line being set on or off. If you enter **YES**, the bit is set on. If you enter **NO** or make no reply, the bit is set off.

Usage Notes

User exit routines are loaded and deleted every time they are called. If your user exit routine is called from a TSP loop and performs I/O or maintains counts, you can load and delete it outside the loop to avoid losing data and to improve performance. You can also use several routines: one routine to perform the open, save the information, and count data in a save area; a second routine to do the I/O; and a final user exit routine to perform the close.

You must store your user exit routines in a load library in LINKLIB or your ISPLLIB concatenation.

For examples, use PMF to look at TSPs BTNTAPRV and BTNTASPT in your base panel data set.

What the Control Line Does

When the TSP is run, the named user exit routine is loaded and called. Any variables you specified on the data or flag field specification panels are available for your user exit's use in their appropriate TSCA fields. These values are reset when your user exit routine ends. The TSCA is the single parameter that is passed to the routine. Upon completion, the routine is deleted.

The USEREXIT TSX Control Line

The format of the USEREXIT control line is:

```
CALL BLGTSX 'USEREXIT',name,argument
```

Note: User exits which require that information be specified on the TSP USEREXIT control line panels (BLM8CU9P,BLM8CU9Q) cannot be run from a TSX. Refer to the Environment section of each of the user exits in "User Exits" on page 263 to determine whether the user exit can be run from a TSX.

Parameter Descriptions

1. name

Valid reply

The 1- to 8-character name of the user exit module.

Default

None

Required

2. argument

Valid reply

A string up to 512 bytes long. The string is loaded into the variable data area (VDA) where the user exit can access it.

Default

None

Optional

Usage Notes and Examples

This is an example of using a USEREXIT control line in a TSX. User exit BLGUSERS is called and parameter BLGUSERS_PARM is passed.

```
CALL BLGTSX 'USEREXIT','BLGUSERS',BLGUSERS_PARM;
```

Return and Reason Codes

The USEREXIT control line does not modify return codes. Therefore, you can code your user exit routine to set the TSCAFRET and TSCAFRES field to any value to communicate with the TSP.

If the user exit routine is not found, Tivoli Information Management for z/OS ABENDs with an 806 ABEND code.

WORDFIX

This control line enables you to repair your database. It can be used for correcting data as well as for manipulating the data for installation-specific purposes. Depending on the information you supply to this control line, you can do any of the following:

- Add an s-word, p-word, or both
- Delete an s-word or p-word
- Change an s-word or p-word
- Change a prefix but not the associated data

- Change the data for a p-word but not the prefix
- Change control information for existing entries. This control data consists of:
  - Whether to cognize the entry
  - Whether and how to perform journaling
  - Replace previous reply setting
  - The panel name collected with the data item.

Note: WORDFIX functions are **only** supported in a TSP. However, the TSX ADDSDATA control line can be used to perform the WORDFIX-like add function and the DELSDATA can be used to perform the WORDFIX-like delete function. In order to perform functions similar to the WORDFIX change data functions in a TSX, the TSX must use the LINK control line to link to a TSP which does the WORDFIX.

Creating a WORDFIX Control Line

You use the following WORDFIX Specification panel to identify the type of change you want to make:

```
BLM8CU9M          WORDFIX CONTROL LINE SUMMARY              PANEL:


    Locate structured word.. __________  Locate prefix........... ______
    Locate word acronym..... __________  Locate validation....... __________
    New structured word..... __________  New prefix.............. ______
    New word acronym........ __________  New validation.......... __________

    Literal data............ ____________________________________
    New data................ ____________________________________
    Use control data........ NO
    Cognize response........ YES
    Cognize only p-word..... NO

  Select one of the choices, or type END to save or CANCEL to discard changes.

                       1. Add entries.
                       2. Delete entries.
                       3. Change s-word.
                       4. Change prefix and/or data.



===>
```

The fields at the top of the panel represent a summary of information contained in the control line and are display-only fields. This information is added to the panel after you create the control line.

- For more information on adding data, see “Adding Data” on page 210.
- For more information on deleting data, see “Deleting Data” on page 217.
- For more information on changing s-word data, see “Changing S-Word Data” on page 221.
- For more information on changing p-word data, see “Changing P-Word Data” on page 225.

General Rules

CAUTION:
You can damage your existing database if you do not use this control line correctly. For information on the security measures you can use to protect against its misuse, refer to the discussion of data integrity and security using TSPs in the *Tivoli Information Management for z/OS Planning and Installation Guide and Reference***.**

Do not specify more than one operation in a single WORDFIX control line. If you do, your data could be lost or damaged.

When specifying more than one WORDFIX control line, insert a new line every time. If you copy previously existing WORDFIX control lines, you can inadvertently carry over information that can cause your new data to be lost.

Do not use the WORDFIX control line to change information (s-word, prefix, or data) collected using the list processor (program exit BLG01396).

The WORDFIX control line is very powerful; use it with caution.The WORDFIX control line validates most changes when the TSP is created. If any of your entries are not valid (such as a nonexistent p-word index), you cannot file the panel. However, some changes do not produce an error, such as changing the s-word for a problem record to a change record.

- The WORDFIX control line works on a single record. You can, however, set up a TSP to loop through multiple records.
- The WORDFIX control line changes or deletes all occurrences of the located data in a record. You can limit this for cases where the data was collected into the record from different panels. When creating the WORDFIX control line, specify a value for the **Locate panel name** field in addition to an s-word or p-word.
- If a located p-word or s-word is marked for deletion, the WORDFIX control line ignores it.
- The WORDFIX control line changes the record in memory, but it does not file the record. You must be in update mode and file the record after the WORDFIX if you want your changes saved in the record.
- When you specify a p-word to be changed or deleted, the value of the **Treat as string data** field must be the same on the specification panel and in the target record. Otherwise, the p-word is not found when the TSP is run.
- When changing an s-word or p-word, unless you specify the **Use control data** field as YES, the existing journal, cognize, replace previous reply, and panel name settings are copied from the original entry.
- When using WORDFIX to add date data, the p-word being added to the record must begin with the characters `DAT`.
- Set the **Skip validation** field to **YES** if you want to make the change without having to consider how the p-word was originally collected or if the user running the TSP has enough authority to have the TSP perform the WORDFIX. Set **Skip validation** to **NO** only if you want to use the assisted-entry panel processor.
- Whenever you want to use variable data, you must specify **Get variable data** as **YES**. If you set the **Use variable data for output** to **YES**, variable data is used as output; otherwise, it is used as the WORDFIX search argument. You cannot use variable data for input and output on the same WORDFIX control line.

- If you want to specify a prefix in the variable data area or **New data** field, set the **Data may contain a prefix** field to **YES**. If you don't set this field, Tivoli Information Management for z/OS interprets the contents of the variable data area or **New data** field as data only.
- Refer to the *Tivoli Information Management for z/OS Panel Modification Facility Guide* for rules regarding prefix restrictions. If you modify prefixes using WORDFIX, the new prefixes must follow these rules.

Adding Data

The following panel is displayed when you select option 1, **Add entries**, from the WORDFIX Specification panel:

```
BLM8CU9U          WORDFIX ADD SPECIFICATION                  PANEL:

Enter 'WORDFIX' add data; cursor placement or input line entry allowed.

 Structured Word Data                      New structured word..... __________
    1. New structured word index..... ____ New word acronym........ __________

 Prefix Word Data
    2. New prefix index.............. ____ New prefix.............. ______
    3. Treat as string data.......... NO_  New validation.......... __________
    4. Get variable data............. NO_
    5. Use variable data for output.. NO_
    6. Data may contain prefix....... ___
    7. New data...................... ________________________________

 Control Data
    8. Use control data...... NO_      11. Cognize response........ YES
    9. New panel name........ ________ 12. Cognize only p-word..... NO_
   10. Replace previous reply YES      13. Journal reply........... YES
                                       14. Journal sequence........ ORDER___
                                       15. Cognize in mixed case?.. NO_
         When you finish, type END to save or CANCEL to discard any changes.

===>
```

Field Descriptions for Panel BLM8CU9U

1. New structured word index

This field indicates the s-word that you want to be added.

Valid reply

The index of an s-word in the dictionary data set, or no reply.

Default

No reply.

2. New prefix index

This field indicates the p-word that you want to add. You can specify the prefix index alone, with variable data obtained from the TSCA, or with the **New data** field.

Prefix index only

If you enter only the **Prefix index** field, it must refer to a prefix that has a literal validation pattern (enclosed by <> delimiters).

Prefix index with variable data

If you specify **Get variable data** as YES, it must refer to a p-word that has no validation pattern associated with it. A user exit routine or a MOVEVAR control line must set up the variable data area and the variable data length prior to processing a WORDFIX control line that specifies this field as **YES**.

Prefix index with literal data

If you enter a prefix index that refers to a p-word without an associated validation pattern, you can specify data in the **New data** field to be added after the prefix.

Valid reply

The index of a p-word that exists in the dictionary data set, or no reply.

Default

No reply.

3. Treat as string data

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want the added p-word entry treated as string data. A string data field is treated as a single-character string during data-entry, rather than as multiple words, and can therefore contain special characters.

For example, the **Description abstract** field of a problem record can contain several words. However, when the **Description abstract** field is designated as a string data field (that is, the **Collect as string** field on the associated assisted-entry panel is set to **YES**), the entire field, including words and spaces between, is treated as one character string.

If you use **YES**, the p-word entry is added as string data.

If you use **NO**, the p-word entry is added as nonstring data.

Default

No reply.

Restrictions

You can use this field only when adding a p-word.

4. Get variable data

This field indicates whether you want this control line to use variable data extracted from the TSCA. The variable data can be additional data added after a prefix if you fill in the **New prefix index** field.

Valid reply

YES, NO, or no reply.

Default

NO

Restrictions

If you enter a prefix index without an associated validation pattern, you must either

enter **YES** in this field or enter data in the **New data** field. If the WORDFIX control line has **YES** in this field, a user exit routine must move data into the variable data area and set the variable data length, or the MOVEVAR control line must move data into the variable data area before processing the WORDFIX control line.

The TSCA contains fields for a pointer to the variable data area (TSCAVDAP) and for the length of that data (TSCAVDAL). The pointer field contains the address of a variable data area that is allocated when the TSP environment is initialized. Your user exit routine must not modify this pointer.

5. Use variable data for output

This field indicates whether you want data in the variable data area to be used for input or output to the WORDFIX control line.

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want data in the variable data area to be used as output.

A reply of **NO** indicates that you want data in the variable data area to be used as input for a search argument.

Default

NO

Restrictions

If you set this field to **YES**, you cannot use the **New data** field.

The **Get variable data** field must be set to **YES**. If **Get variable data** is set to **NO**, this field is ignored.

6. Data may contain prefix

This field indicates whether either the variable data area or **New data** field contains a prefix.

Valid reply

YES, NO, or no reply.

If this field is set to **YES** and the data contains a slash (/) or underscore (_) character in the first six positions, then the data is assumed to contain a prefix.

If this field is set to **NO**, the data is assumed to contain only prefix data (no prefix).

Default

NO

Restrictions

If you set this field to **YES**, you must specify either of the following:

- Set **Get variable data** to **YES** and set **Use variable data for output** to **YES**.

  You must also have set up the variable data area and moved data into it before this control line is run. You can do this earlier in your TSP by calling a user exit routine that sets the appropriate TSCA fields or by using a MOVEVAR control line.

- Enter data in the **New data** field.

7. New data

This field indicates the data that you want to associate with the prefix. The data that is entered in this field is collected in the case entered by the user.

Valid reply

A string of 1 to 32 characters or no reply.

Default

No reply.

Restrictions

If you enter data in this field, you cannot set the **Use variable data for output** field to **YES**.

When an SBCS comma is required as the first, or only, character of this field, you must precede the SBCS comma with an SBCS space character.

8. Use control data

The value of this field is ignored on this panel. When you add data, control data is always used.

9. New panel name

Valid reply

An 8-character panel name, or no reply.

A reply in this field indicates that you want the name of a panel associated with the added data.

Default

No reply.

Restrictions

The panel name must consist of SBCS characters only.

10. Replace previous reply

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want duplicate entries for the same field to be replaced in the record before it is filed.

A reply of **NO** indicates that you want to preserve duplicate entries for the same field. This is useful for panels that collect open-ended lists of information.

Default

YES

11. Cognize response

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want the data to become searchable in the database.

A reply of **NO** means that users cannot search for this information.

Default

YES

12. Cognize only p-word

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want only p-words to become searchable in the database.

A reply of **NO** indicates that you want both p-words and s-words to become searchable.

Default

NO

Restrictions

The **Cognize response** field must be set to YES. This field is ignored if the **Cognize response** field is set to **NO**.

13. Journal reply

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want the value of the added data (p-word or data) recorded in the database history when the record is filed. If multiple values are assigned to the same p-word or s-word before the record is filed, only the last value assigned is recorded.

A reply of **NO** indicates that you do not want added data recorded in the database history.

Default

NO

Restrictions

If you set this field to **YES**, you must also set the **Cognize response** field to **YES**.

14. Journal sequence

Valid reply

FIRST, ORDER, or no reply.

A reply of FIRST indicates that you want the entry to appear first in the history for a given update.

A reply of ORDER indicates that you want the entry to appear in the order it which it occurs in the record.

Default

ORDER

15. Cognize in mixed case?

Valid reply

YES, NO, or no reply.

This defines how the data is cognized, that is, stored in the SDIDS for searching.

A reply of **YES** indicates that the data is cognized exactly as it is stored in the database; to find this record, a search argument must be case-specific and both the letter and the case must match. For example, Open will find only Open, but neither OPEN nor open.

A reply of **NO** indicates that the data is cognized in all uppercase; to find this record, a case-insensitive search argument can have letters in any case. For example, Open will find Open, OPEN, open, oPeN, OpEn, and so forth.

Default

NO

New structured word

If you enter data in the **New structured word index** field when you create the control line, this field is filled in automatically.

New word acronym

If you enter data in the **New structured word index** field when you create the control line, this field is filled in automatically. It is part of the s-word entry in the dictionary. However, this information is not used. This field could be blank if a word acronym does not exist for the entry.

New prefix

If you enter data in the **New prefix index** field when you create the control line, this field is filled in automatically. It displays the prefix found in the dictionary.

New validation

If you enter data in the **New prefix index** field when you create the control line, this field is filled in automatically. It displays the validation pattern associated with the prefix index.

What This Selection Does

This selection is functionally equivalent to an ADD control line in a control panel. You can add an s-word, p-word, or both, depending upon the needs of your installation. The advantages that this selection has over control panels is that you can specify variable data and that you do not necessarily have to add an entry to your dictionary.

To add an s-word, specify the index of the s-word in the **New structured word index** field and change any control data settings that you do not want (only **Cognize reply** and **New panel name** apply to s-words).

To add a p-word, you can specify data in any of several ways. Assuming you want a prefix, you can specify any of the following:

- Prefix index only. If you want to add a p-word that is in the dictionary and you have its index, specify a value for **New prefix index**.
- Literal data only. If you have no index for, or don't want to bother looking for, a dictionary entry for the p-word you want to add, enter the complete p-word into the **New data** field and set **Data may contain prefix** to **YES**.

- Variable data only. If your TSP puts p-word information into the variable data area, set **Get variable data** to YES, **Use variable data for output** to **YES**, and **Data may contain prefix** to **YES**. A user exit routine or a MOVEVAR control line must set up the variable data area and the variable data length before this WORDFIX control line can be processed.
- Prefix with literal data specified separately. Enter the prefix part of the p-word into the **New prefix index** field, and put the p-word value into the **New data** field.
- Prefix with variable data specified separately. Enter the prefix part of the p-word into the **New prefix index** field, and get the p-word value from the variable data area. (Your TSP must have previously put this value into the variable data area.)

You must also set the control data fields the way you want them added to your record. Regardless of how the **Use control data** field is set, the control data fields are always used.

Examples of Using WORDFIX to Add Data

Suppose you decide to start using the Integration Facility, but you want to continue using your existing privilege class records. You can add the s-word that identifies the special Integration Facility privilege class to your own privilege class records. This example uses the Problem Controller privilege class. The s-word index is 7D34.

1. Write a one-line TSP that contains a WORDFIX control line with the following fields filled in:
   - **New structured word index** set to 7D34
   - **New prefix index** set to 0072 (AUTH/YES). Alternatively, you could specify AUTH/YES in the **New data** field.
2. Issue the UPDATE command to update your privilege class record.
3. Issue the RUN command, specifying the name of the TSP you just created.
4. Issue the VIEW INTERNALS command to verify that the data actually got added to the record.
5. File the record to permanently save the data you just added to the record.

Suppose your users tell you that when a record is closed as a duplicate of another record, they want the record ID of the duplicate record added into the record that was identified as being the original problem. You can run a job every night in the Master privilege class that does a TSP search for all closed records that were last updated on that day and have a closing code of DUP. Your TSP can also do a block display of all records in the search results list and perform the following processing:

1. MOVEVAR TSCACRID with **Replace data** set to **YES** to move the ID of the current record into the variable data area
2. SETFIELD TSCAUFLD with **Get variable data** set to **YES**
3. FINDSDATA of RNPD/. to find the original problem number
4. ADDDATA of UPDATE R (the command for updating a record)
5. MOVEVAR TSCASDF with **Replace data** set to YES to move the record ID that FINDSDATA found
6. ADDDATA with **Get variable data** set to YES to move the update request into the command line reply buffer

7. PROCESS to actually perform the update
8. MOVEVAR TSCAUFLD with **Replace data** set to **YES**
9. WORDFIX with:
   - **New structured word index** set to 8000 (an s-word you added for duplicate record IDs)
   - **New prefix index** set to 8000 (a p-word you added that has a prefix of DUP/ and no data)
   - **Get variable data** set to **YES** (the record ID for the record closed as DUP)
   - **Use variable data for output** set to **YES**.
10. ADDDATA with **Literal data** set to 9 (or whatever selection files the record)
11. PROCESS to actually file the record.

Deleting Data

The following panel is displayed when you select option 2, **Delete entries**, from the WORDFIX Specification panel:

```
BLM8CU9V          WORDFIX DELETE SPECIFICATION              PANEL:

Enter 'WORDFIX' delete data; cursor placement or input line entry allowed.

 Structured Word Data
    1. Locate structured word index.. ____ Locate structured word.. __________
                                           Locate word acronym..... __________
 Prefix Word Data
    2. Locate prefix index........... ____ Locate prefix........... ______
                                           Locate validation....... __________
    3. Literal data.................. _____________________________________
    4. Treat as string data.......... NO_
    5. Get variable data............. NO_
    6. Skip validation............... NO_


        When you finish, type END to save or CANCEL to discard any changes.




===>
```

Note: Do not use the WORDFIX control line to delete any information collected using the list processor (program exit BLG01396).

Field Descriptions for Panel BLM8CU9V

1. Locate structured word index

This field indicates the structured description entries that you want deleted. If you enter the s-word index, all structured description entries in the current record that contain the associated s-word become candidates for deletion.

Valid reply

The index key of an s-word in the dictionary data set, or no reply.

Default
No reply.

Restrictions
If you fill in this field, you cannot use the **Locate prefix index** field, set **Get variable data** to YES, or specify a value for the **Literal data** field.

2. Locate prefix index

This field indicates the index of the p-word you want deleted. You can use the prefix index alone, with variable data obtained from the TSCA, or with the **Literal data** field. A list of conditions for using this field follows.

Prefix index only

If you enter only the **Prefix index** field, it must refer to a prefix that has a literal validation pattern (enclosed by <> delimiters).

Prefix index with variable data

If you specify **Get variable data** as YES, it must refer to a p-word that has no validation pattern associated with it. A user exit routine or a MOVEVAR control line must set up the variable data area and the variable data length before processing a WORDFIX control line that specifies this field as **YES**. The resulting p-word is the search word used to locate an entry in the record.

Prefix index with literal data

If you enter a prefix index that refers to a p-word without an associated validation pattern, you can specify data in the **Literal data** field to be added after the prefix. The resulting p-word is the search word used to locate an entry in the record.

If you enter the prefix index, all SDEs in the current record that contain the complete prefix become candidates for deletion.

Valid reply
The index key of a p-word in the dictionary data set, or no reply.

Default
No reply.

Restrictions
If you use this field, you cannot use the **Locate structured word index** field.

3. Literal data

Valid reply
A string of 1 to 32 characters, or no reply.

The data in this field is used to locate an entry. You can enter data to be used with the **Locate prefix index** field to specify a p-word, or you can enter the entire p-word search word (for example, PERS/NEWDATA). The data that is entered in this field is collected in the case entered by the user.

Default
No reply.

Restrictions
If you specify a **Locate prefix index** field that does not contain an associated

validation pattern, you must either enter YES in the **Get variable data** field, or this field must contain data. You cannot fill in this field if data is entered in the **Locate structured word index** field or if you entered YES in the **Get variable data** field.

When an SBCS comma is required as the first, or only, character of the **Literal data** field, you must precede the SBCS comma with an SBCS space character.

4. Treat as string data

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want the search word treated as string data, so only string data fields in the record are found. A string data field is treated as a single-character string during data-entry, rather than as multiple words, and can therefore contain special characters.

For example, the **Description abstract** field of a problem record can contain several words. However, when this field is designated as a string data field (that is, the **Collect as string** field on the associated assisted-entry panel is set to **YES**), the entire field, including words and spaces between, is treated as one character string.

If you use **YES**, the entire string must be specified. If you are using non-Latin translation tables, no global characters (* .) can be used to search for string data.

If you use **NO**, only nonstring data is searched for.

Default

No reply.

Restrictions

You can use this field only when the locate argument is a p-word.

5. Get variable data

This field indicates whether you want this control line to use variable data extracted from the TSCA. The variable data can be data added after a prefix if you have set the **Locate prefix index** field, or the actual prefix search word used to locate data items to be changed.

Valid reply

YES, NO, or no reply.

Default

NO

Restrictions

If you enter a prefix index without an associated validation pattern, you must either enter **YES** in this field or enter data in the **Literal data** field. You cannot use this field if you use the **Locate structured word index** field. If the WORDFIX control line has **YES** in this field, a user exit routine must move data into the variable data area and set the variable data length, or the MOVEVAR control line must move data into the variable data area before processing the WORDFIX control line.

The TSCA contains fields for a pointer to the variable data area (TSCAVDAP) and for the length of that data (TSCAVDAL). The pointer field contains the address of a variable data area that is allocated when the TSP environment is initialized. Your user exit routine must not modify this pointer.

6. Skip validation

Set this field to **YES** to delete data. If this field is set to **NO** or left blank, Tivoli Information Management for z/OS processes this control line as a change if the located data was originally collected by an assisted-entry panel.

Locate structured word

If you enter data in the **Locate structured word index** field when you create the control line, this field is filled in automatically. It displays the s-word found in the dictionary.

Locate word acronym

If you enter data in the **Locate structured word index** field when you create the control line, this field is filled in automatically. It is part of the s-word entry in the dictionary. However, this field is not used. This field could be blank if a word acronym does not exist for the entry.

Locate prefix

If you enter data in the **Locate prefix index** field when you create the control line, this field is filled in automatically. It displays the prefix found in the dictionary.

Locate validation

If you enter data in the **Locate prefix index** field when you create the control line, this field is filled in automatically. It displays the validation pattern associated with the prefix index.

What This Selection Does

This selection is used to delete data you no longer want in your record. Maybe you decided that a field is unnecessary or maybe you needed to add something temporarily but you don't want it filed with the record. In either case, you can delete any s-word or p-word that you want.

Note: If you want to delete the s-word and p-word for a single data item, you need two WORDFIX control lines. Delete the p-word first, then delete the s-word. If you delete a data item containing an s-word, Tivoli Information Management for z/OS checks the data item immediately preceding the one being deleted to determine if it can also be deleted.

To delete an s-word, specify a value for **Locate structured word index**.

For deleting a p-word, you have a choice. Assuming the item has a prefix, you can specify any of the following:

- Prefix index only. If you know the prefix index, specify **Locate prefix index**.
- Literal data only. If you don't have, or don't want to bother looking for, a dictionary entry for the p-word to be deleted, specify a value for the **Literal data** field. This field is especially useful because you can specify the truncation character (.) for the data. So, for example, if you want to delete the **Reporter department** field from the record, you can specify GROS/. in this field.

- Variable data only. If your TSP builds the search argument in the variable data area, set **Get variable data** to **YES**.
- Prefix index and literal data specified separately. Set the **Locate prefix index** field to the prefix index, and set the **Literal data** field to the data part of the p-word.
- Prefix index and variable data specified separately. Set the **Locate prefix index** field to the prefix index. Your TSP must set up the variable data area and variable data length and move data into the variable data area. This can be done either with a user exit routine or with a MOVEVAR control, but it must occur before this WORDFIX control line is run.

Note: Be sure to create a new WORDFIX control line when you want to delete data. If the **Use control data** field remains set to **YES** on another WORDFIX specification panel, Tivoli Information Management for z/OS changes the control data in the located entry instead of deleting it.

Examples of Using WORDFIX to Delete Data

Suppose you find out that your users mostly perform freeform searches. To minimize the size of the SDIDS, you decide to delete entries for the selections your users make from the Problem Summary panel (Reporter data, Status data, and so on). You change panel BLG0BU00 to collect function code index 000A for each of the selections, and now you want to delete the corresponding information in your database.

Write a TSP to search for all problem records and do a block update of all records on the search results list. Then add the following control lines:

1. WORDFIX with **Locate structured word index** set to 0CFC (Reporter data)
2. ADDDATA with **Literal data** set to 9 (or whatever selection files a record from the Problem Summary panel)
3. PROCESS to actually file the record.

Suppose you have deleted the **Reporter department** field from all your panels, and now you want to delete all entries in this field from your database. Write a TSP to search for all the problem records containing a reporter department (GROS/.) and to do a block update of all the records on the search results list. Then add the following control lines:

1. WORDFIX with **Literal data** set to GROS/. and **Skip validation** set to **YES** to delete the p-word
2. WORDFIX with **Locate structured word index** set to 0B9B to delete the s-word
3. ADDDATA with **Literal data** set to 9 (or whatever selection files a record from the Problem Summary panel)
4. PROCESS to actually file the record.

Changing S-Word Data

The following panel is displayed when you select option 3, **Change s-word**, from the WORDFIX Specification panel:

```
BLM8CU9W          WORDFIX S-WORD SPECIFICATION              PANEL:

Enter 'WORDFIX' S-word data; cursor placement or input line entry allowed.

 Structured Word Data
    1. Locate structured word index.. ____ Locate structured word.. __________
                                           Locate word acronym..... __________
    2. New structured word index..... ____ New structured word..... __________
                                           New word acronym........ __________

    3. Locate panel name.............. ________

 Control data
   4. Use control data...... NO_        7. Cognize response........ YES
   5. New panel name........ ________   8. Cognize only p-word..... NO_
   6. Replace previous reply YES

         When you finish, type END to save or CANCEL to discard any changes.




===>
```

General Rule

Do not use the WORDFIX control line to change s-words collected using the list processor (program exit BLG01396).

Field Descriptions for Panel BLM8CU9W

1. Locate structured word index

This field indicates the structured description entries (SDEs) that you want changed. If you enter the s-word index, all SDEs in the current record that contain the associated s-word become candidates for replacement.

Valid reply

The index of an s-word in the dictionary data set, or no reply.

Default

No reply.

2. New structured word index

This field indicates an s-word that replaces the s-word in the located data items. If you enter the s-word index, all occurrences of the associated s-word word replace all occurrences of the old s-word in the record. You can use the **Locate panel name** field to limit the data items that are changed.

Valid reply

The index of an s-word in the dictionary data set, or no reply.

Default

No reply.

Restrictions

If you fill in this field, you must also fill in the **Locate structured word index** field.

3. Locate panel name

Valid reply

An 8-character panel name or no reply.

Default

No reply.

Restrictions

The panel name must consist of SBCS characters only.

4. Use control data

Valid reply

YES or **NO**.

A reply of **YES** indicates that you want the values in the following fields:

New panel name
Replace previous reply
Cognize response
Cognize only p-word

to be associated with the data being changed.

Default

NO

Restrictions

None.

5. New panel name

Valid reply

An 8-character panel name or no reply. A reply in this field indicates that you want the name of a panel associated with the changed data.

Default

No reply.

Restrictions

The panel name must consist of SBCS characters only.

6. Replace previous reply

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want duplicate entries for the same field to be replaced in the record before it is filed.

A reply of **NO** indicates that you want to preserve duplicate entries for the same field. This is useful for panels that collect open-ended lists of information.

Default

YES

7. Cognize response

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want the data to become searchable in the database.

A reply of **NO** means that users cannot search for this information.

Default

YES

8. Cognize only p-word

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want only p-words to become searchable in the database.

A reply of **NO** indicates that you want both p-words and s-words to be searchable.

Default

NO

Restrictions

The **Cognize response** field must be set to YES. This field is ignored if the **Cognize response** field is set to **NO**.

Locate structured word

If you enter data in the **Locate structured word index** field when you create the control line, this field is filled in automatically. It displays the s-word found in the dictionary.

Locate word acronym

If you enter data in the **Locate structured word index** field when you create the control line, this field is filled in automatically. It is part of the s-word entry in the dictionary. However, it is not used for the search.

New structured word

If you enter data in the **New structured word index** field when you create the control line, this field is filled in automatically.

New word acronym

If you enter data in the **New structured word index** field when you create the control line, this field is filled in automatically. It is part of the s-word entry in the dictionary. However, it is not used for the search. This field could be blank if a word acronym does not exist for the entry.

What This Selection Does

This function is used to change one s-word to another, to change control information associated with an s-word, or both.

To change an s-word, specify the index of the s-word you want changed in the **Locate structured word index** field. Specify the index to which you want it changed in the **New structured word index** field.

To change control data without changing an s-word, specify the index of the s-word whose control data you want changed. Then specify **YES** for **Use control data** and update the appropriate control data.

Note: Only **New panel name**, **Replace previous reply**, and **Cognize reply** are valid for s-word changes. Any other control data you change is ignored.

To change both an s-word and control data, specify the index of the s-word you want changed in the **Locate structured word index** field. Specify the index to which you want it changed in the **New structured word index** field. Then specify **YES** for **Use control data**, and update the appropriate control data.

Note: Only **New panel name**, **Replace previous reply**, and **Cognize reply** are valid for s-word changes. Any other control data you change is ignored.

Examples of Using WORDFIX to Change S-Words or Control Data

Suppose you decide to uncognize some of your selections to improve database performance. You decide to start with resolver data for problem records.

Write a TSP to find all problem records. Add control lines to do a block update of all the records on the search results list. Then add the following control lines:

1. WORDFIX with the following:
   - **Locate structured word index** set to 0C5E (the s-word for resolver data)
   - **Use control data** set to **YES** (so you can change the cognize setting)
   - **Cognize reply** set to **NO**
2. ADDDATA with **Literal data** set to 9 (or whatever selection files the record)
3. PROCESS to actually file the record.

Changing P-Word Data

The following panel is displayed when you select option 4, **Change prefix and/or data**, from the WORDFIX Specification panel:

```
BLM8CU9Y           WORDFIX P-WORD SPECIFICATION              PANEL:

Enter 'WORDFIX' P-word data; cursor placement or input line entry allowed.

Locate data
    1. Locate prefix index.......... ____ Locate prefix.......... ______
    2. Get variable data............ NO_  Locate validation....... __________
    3. Literal data................. ____________________________________
    4. Locate panel name............ ________
    5. Treat as string data......... NO_
New data
    6. New prefix index............. ____ New prefix.............. ______
    7. Use variable data for output.. NO_  New validation.......... __________
    8. Data may contain prefix...... ___
    9. Skip validation.............. NO_
   10. New data..................... ____________________________________
 Control Data
   11. Use control data...... NO_      14. Cognize response........ YES
   12. New panel name........ ________ 15. Cognize only p-word..... NO_
   13. Replace previous reply YES     16. Journal reply........... YES
                                      17. Journal sequence........ ORDER___
                                      18. Cognize in mixed case?.. NO_
        When you finish, type END to save or CANCEL to discard any changes.

===>
```

General Rule

Do not use the WORDFIX control line to change p-words collected using the list processor (program exit BLG01396).

When changing data for a date or time field, remember that date and time values are stored in the record in both internal format and external format. In addition, if the field is a Universal Time field, the date or time is also stored in UT, though with a slightly different prefix. To properly change the value using WORDFIX, you must change ALL of the formats. It is strongly recommended that you use validation when changing the data because the validation process will update all forms of the date or time. If you choose not to validate, you will usually need multiple WORDFIX control lines, one control line for each form of the value.

When changing only the prefix for a date or time value which is defined as a Universal Time field, you must change the special Universal Time prefix as well. This will require a second WORDFIX control line. If you do not do this, the date or time value might be processed incorrectly in later operations.

Field Descriptions for Panel BLM8CU9Y

1. Locate prefix index

This field indicates the index of the p-word you want changed. You can use the prefix index alone, with variable data obtained from the TSCA, or with the **Literal data** field.

Prefix index only

If you enter only the **Prefix index** field, it must refer to a prefix that has a literal validation pattern (enclosed by <> delimiters).

Prefix index with variable data

If you specify **Get variable data** as YES, it must refer to a p-word that has no validation pattern associated with it. A user exit routine or a MOVEVAR control line must set up the variable data area and the variable data length prior to processing a WORDFIX control line that specifies this field as **YES**. The resulting p-word is the search word used to locate an entry in the record.

Prefix index with literal data

If you enter a prefix index that refers to a p-word without an associated validation pattern, you can specify data in the **Literal data** field to be added after the prefix. The resulting p-word is the search word used to locate an entry in the record.

If you enter the prefix index, all SDEs in the current record that contain the complete prefix become candidates for replacement or deletion. You can use the **Panel name** field to limit the data items that are to be replaced or deleted.

Valid reply

The index key of a p-word in the dictionary data set, or no reply.

Default

No reply.

2. Get variable data

This field indicates whether you want this control line to use variable data extracted from the TSCA. The variable data can be additional data added after a prefix if you fill in the **Locate prefix index** field, or the actual prefix search word used to locate data items to be changed.

Valid reply

YES, NO, or no reply.

Default

NO

Restrictions

If you enter a prefix index without an associated validation pattern, you must either enter **YES** in this field or enter data in the **Literal data** field. If the WORDFIX control line has **YES** in this field, a user exit routine must move data into the variable data area and set the variable data length or the MOVEVAR control line must move data into the variable data area before processing the WORDFIX control line.

The TSCA contains fields for a pointer to the variable data area (TSCAVDAP) and for the length of that data (TSCAVDAL). The pointer field contains the address of a variable data area that is allocated when the TSP environment is initialized. Your user exit routine must not modify this pointer.

3. Literal data

Valid reply

A string of 1 to 32 characters, or no reply.

The data entered in this field is used to locate an entry to be replaced. You can enter data that you want added after a prefix, specified by the **Locate prefix index** field, or enter the entire p-word search word. The data that is entered in this field is collected in the case entered by the user.

Default
No reply.

Restrictions
If you specify a **Locate prefix index** field that does not contain an associated validation pattern, you must either enter YES in the **Get variable data** field or enter data in this field. You cannot fill in this field if you set the **Get variable data** field to **YES** and **Use variable data for output** to **NO**.

If you want to enter an SBCS comma as the first, or only, character in the **Literal data** field, you must precede the SBCS comma with an SBCS space character.

4. Locate panel name

Valid reply
An 8-character panel name or no reply.

Default
No reply.

Restrictions
The panel name must consist of SBCS characters only.

5. Treat as string data

Valid reply
YES, NO, or no reply.

A reply of **YES** indicates that you want the search word treated as string data, so only string data fields in the record are found. A string data field is treated as a single character string during data-entry, rather than as multiple words, and can therefore contain special characters.

For example, the **Description abstract** field of a problem record can contain several words. However, when this field is designated as a string data field (that is, the **Collect as string** field on the associated assisted-entry panel is set to **YES**), the entire field, including words and spaces between, is treated as one character string.

If you use **YES**, the entire string must be specified. If you are using non-Latin translation tables, no global characters (* .) can be used to search for string data.

If you use **NO**, only nonstring data is searched for.

Default
No reply.

Restrictions
None.

6. New prefix index

If the data that you are trying to locate was collected other than by an assisted-entry panel or if **Skip validation** is set to YES, this field indicates the index of a p-word that is to replace the prefix entered in the **Locate prefix index** field. Enter a value in this field if you want to change the current prefix to a new prefix.

If the new prefix index has a validation pattern that is completely enclosed by delimiter symbols (<>), the data between the symbols replaces the data associated with the old prefix.

If the new prefix index does not have a validation pattern or its validation pattern has a literal validation pattern, the data associated with the new prefix is the same as the data associated with the old prefix.

Valid reply

The index of a p-word that exists in the dictionary data set, or no reply.

Default

No reply.

Restrictions

If the located data was collected by an assisted-entry panel or if **Skip validation** is set to **NO**, this field is ignored.

7. Use variable data for output

This field indicates whether you want data in the variable data area to be used for input or output to the WORDFIX control line.

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want data in the variable data area to be used as output.

A reply of **NO** indicates that you want data in the variable data area to be used as input.

Default

NO

Restrictions

If this field is set to **YES**, you cannot use the **New data** field. The **Get variable data** field must be set to **YES**. If **Get variable data** is set to **NO**, this field is ignored.

8. Data may contain prefix

This field indicates whether either the variable data area or **New data** field contains a prefix.

Valid reply

YES, NO, or no reply.

If this field is set to **YES** and the data contains a slash (/) or underscore (_) character in the first six positions, then the data is assumed to contain a prefix.

If this field is set to **NO**, the data is assumed to contain only prefix data (no prefix).

Default

NO

Restrictions

If you set this field to **YES**, you must specify either of the following:

- Set **Get variable data** to **YES** and set **Use variable data for output** to **YES**.

  You must also have set up the variable data area and moved data into it before this control line is run. You can do this earlier in your TSP by calling a user exit routine that sets the appropriate TSCA fields or by using a MOVEVAR control line.

- Enter data in the **New data** field.

9. Skip validation

This field indicates whether you want data to be validated through the assisted-entry panel before it is changed.

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you do *not* want data validated before it is changed.

A reply of **NO** indicates that you want data validated. However, the control data on the assisted-entry panel is used for the change, not any control information on the WORDFIX control line.

Default

NO

Restrictions

If the data to be changed was collected by a panel type other than assisted-entry, this field is ignored.

10. New data

This field indicates the data that you want to replace the existing data presently located in the SDEs. Enter data exactly as it is to be processed by the panel in the located data items. This data is a response to an assisted-entry panel if **Skip validation** is set to **NO**. If you do not fill in the **New data** field, the current response in the data item is used to reply to the panel and to subsequently replace the located entry. Use this field to change multiple records when they contain the same field, such as a person's phone number or department.

Valid reply

A string of 1 to 32 characters or no reply.

Default

No reply.

Restrictions

If you want to enter an SBCS comma as the first, or only, character in this field, you must precede the SBCS comma with an SBCS space character.

If you use this field, you cannot set **Get variable data** to **YES** or **Use variable data for output** to **YES**.

11. Use control data

Valid reply

YES or **NO**.

A reply of **YES** indicates that you want the values in the following fields:

Treat as string data
New panel name
Replace previous reply
Cognize response
Cognize only p-word
Journal reply
Journal sequence

to be associated with the data being added.

Default

NO

12. New panel name

Valid reply

An 8-character panel name or no reply. A reply in this field indicates that you want the name of a panel associated with the added data.

Default

BLGTSADD

Restrictions

The panel name must consist of SBCS characters only.

13. Replace previous reply

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want duplicate entries for the same field to be replaced in the record before it is filed.

A reply of **NO** indicates that you want to preserve duplicate entries for the same field. This is useful for panels that collect open-ended lists of information.

Default

YES

14. Cognize response

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want the data to become searchable in the database.

- If the **Cognize only p-word** field is set to NO, both the s-word and p-word are cognized.
- If the **Cognize only p-word** field is set to YES, the s-word is uncognized, but the p-word stays cognized.

A reply of **NO** means that you do not want users to be able to search for this information. Neither the s-word nor the p-word is cognized.

Default

YES

15. Cognize only p-word

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want only p-words to become searchable in the database.

A reply of **NO** indicates that you want both p-words and s-words to become searchable.

Default

NO

Restrictions

The **Cognize response** field must be set to YES. This field is ignored if the **Cognize response** field is set to **NO**.

16. Journal reply

Valid reply

YES, NO, or no reply.

A reply of **YES** indicates that you want the value of the added data (p-word or p-word) recorded in the database history when the record is filed. If multiple values are assigned to the same p-word or s-word before the record is filed, only the last value assigned is recorded.

A reply of **NO** indicates that you do not want added data recorded in the database history.

Default

NO

Restrictions

If you set this field to **YES**, you must also set the **Cognize response** field to **YES**.

17. Journal sequence

Valid reply

FIRST, ORDER, or no reply.

A reply of FIRST indicates that you want the entry to appear first in the history for a given update.

A reply of ORDER indicates that you want the entry to appear in the order it which it occurs in the record.

Default

ORDER

18. Cognize in mixed case?

Valid reply

YES, NO, or no reply.

This defines how the data is cognized, that is, stored in the SDIDS for searching.

A reply of **YES** indicates that the data is cognized exactly as it is stored in the database; to find this record, a search argument must be case-specific and both the letter and the case must match. For example, Open will find only Open, but neither OPEN nor open.

A reply of **NO** indicates that the data is cognized in all uppercase; to find this record, a case-insensitive search argument can have letters in any case. For example, Open will find Open, OPEN, open, oPeN, OpEn, and so forth.

Default

NO

Locate prefix

If you enter data in the **Locate prefix index** field when you create the control line, this field is filled in automatically. It displays the prefix found in the dictionary.

Locate validation

If you enter data in the **Locate prefix index** field when you create the control line, this field is filled in automatically. It displays the validation pattern associated with the prefix index.

New prefix

If you enter data in the **New prefix index** field when you create the control line, this field is filled in automatically. It displays the prefix found in the dictionary.

New validation

If you enter data in the **New prefix index** field when you create the control line, this field is filled in automatically. It displays the validation pattern associated with the prefix index.

What This Selection Does

This function is used for several purposes. You can:

- Change a prefix, but not the associated data
- Change the data, but not the prefix
- Change both prefix and data
- Change control data for a p-word.

Note: If you set **Skip validation** to NO, Tivoli Information Management for z/OS drives your change through the assisted-entry panel processor. Because of this, you cannot change the prefix or the control data with WORDFIX; you must do it by changing the panel itself in PMF.

To change a p-word, you can specify what you want changed in any of several ways:

- Prefix index only. If you know the prefix index in the dictionary, specify **Locate prefix index**.
- Literal data only. If you don't have, or don't want to bother looking for, a dictionary entry for the p-word to be changed, specify both prefix and data in the **Literal data** field.
- Variable data only. If your TSP builds the search argument in the variable data area, set **Get variable data** to **YES**.
- Prefix index and literal data specified separately. Set the **Locate prefix index** field to the prefix index and the **Literal data** field to the data part of the p-word.
- Prefix index and variable data specified separately. Set the **Locate prefix index** field to the prefix index. Your TSP must set up the variable data area and variable data length and move data into the variable data area. This can be done either by user exit routine or with a MOVEVAR control, but it must occur before this WORDFIX control line is run.

You must also specify new values. How to do this depends upon what you want changed.

Changing Prefixes Only

- With prefix index only. If you know the prefix index in the dictionary, specify a value for the **New prefix index** field.
- With literal data only. If you don't have, or don't want to bother looking for, a dictionary entry for the new p-word, specify a value for the **New data** field and set the **Data may contain prefix** field to **YES**.
- With variable data only. If your TSP loads the new prefix into the variable data area, set **Get variable data** to **YES**, set **Use variable data for output** to **YES**, and set **Data may contain prefix** to **YES**.

Changing Data Only

In this case, the **New prefix index** field must point to a p-word containing validation data but no prefix. As an alternative, you could set this field to 0000. Then, specify the new data in one of the following ways:

- With literal data. Enter the changed data into the **New data** field. Set **Use variable data for output** to **NO**.
- With variable data. Set **Get variable data** to **YES** and **Use variable data for output** to YES.

Changing Both Prefix and Data

- With prefix index only. Set **New prefix index** to point to a p-word with both prefix and validation data. Make sure **Use variable data for output** is set to NO. Make sure **New data** is blank.
- With literal data only. Set **New prefix index** to 0000. Make sure **Use variable data for output** is set to **NO**. Enter both prefix and validation data into the **New data** field and set **Data may contain prefix** to **YES**.
- With variable data only. Set **Get variable data** to **YES**. Set **New prefix index** to 0000. Set **Use variable data for output** to **YES**. Set **Data may contain prefix** to **YES**. Make sure that **New data** is blank.

  Your TSP must set up the variable data area and variable data length and have both prefix and validation data values loaded into the variable data area. This can be done either with a user exit routine or a MOVEVAR control line, but it must be done before this WORDFIX control line is run.
- With prefix index and literal data. Set **New prefix index** to point to a p-word that contains only a prefix. Enter the validation data into the **New data** field. Make sure that **Use variable data for output** is set to **NO**.
- With prefix index and variable data. Set **New prefix index** to point to a p-word that contains only a prefix. Set **Get variable data** to **YES** and **Use variable data for output** to **YES**. Make sure **New data** is blank.

  Your TSP must set up the variable data area and variable data length and have the validation data value loaded into the variable data area. This can be done either with a user exit routine or a MOVEVAR control line, but it must be done before this WORDFIX control line is run.

Changing Control Data

To change control data without changing a p-word, specify the p-word for which you want the control information changed. Then, set **Use control data** to **YES** and change the appropriate control data. Set **Skip validation** to **YES** so that the control information you specified is used.

To change a p-word and its control data at the same time, specify both p-words as described previously. Set **Use control data** to YES and update the appropriate control information. Set **Skip validation** to **YES** so that the control information you specified is used.

Examples of Using WORDFIX to Change P-Words or Control Data

Changing Prefixes Only

Suppose you decide to create a new status for change records called CSTAC/. You have existing change records that use the STAC/ prefix.

Update your change panel flow to go to a new assisted-entry panel when the user enters a status on a data-entry panel. To do this, add new dictionary entries for your new prefix as well as add an entry for CSTAC/ with no validation data. Suppose the p-word index for this is 8001.

Write a TSP to find all change records. Add control lines to do a block update of all the records on the search results list. Then add the following:

1. WORDFIX with:
   - **Literal data** set to STAC/. (to find any status)
   - **New prefix index** set to 8001
   - **Skip validation** set to **YES**
2. ADDDATA with **Literal data** set to 9 (or whatever selection files the record)
3. PROCESS to actually file the record.

Because you want the prefix changed but the data left alone, the WORDFIX control line is built with a new prefix but no new data. This signals Tivoli Information Management for z/OS to use the data from the located p-word.

Changing Data Only

Suppose you decide to create a new status for problem records that have been fixed but not yet closed. Update the dictionary and problem status assisted-entry panel to contain STAC/FIXED.

Write a TSP to find all problem records with a status of OPEN that contain resolution data. Add control lines to do a block update of all the records on the search results list. Then add the following:

1. WORDFIX with:
   - **Literal data** set to STAC/. (to find any status)
   - **Skip validation** set to **YES**
   - **New data** set to FIXED
2. ADDDATA with **Literal data** set to 9 (or whatever selection files the record)

3. PROCESS to actually file the record.

Because you want the data changed but the prefix left alone, the WORDFIX control line is built with new validation data but no new prefix. This signals Tivoli Information Management for z/OS to use the prefix from the located p-word.

Usage Notes

When using the WORDFIX control line to repair a record, you must use the UPDATE command to make the record current so Tivoli Information Management for z/OS can file the record. However, when you are testing a TSP that uses a WORDFIX control line, you can use the DISPLAY command to change the record.

After the WORDFIX control line is run, you can use the VIEW INTERNAL command to see the results. If WORDFIX worked correctly, you find the items you fixed and don't find the items you deleted. If not, you might have set up the wrong search or entered the data incorrectly. When you are satisfied that WORDFIX is working correctly, change the TSP to update rather than display the record. Be sure to test the TSP again to make sure the record is being filed properly.

When fixing assisted-entry panel entries, always use a prefix to locate the data, not an s-word. Even if you are changing the only s-word on the panel, your margin of error is smaller when you use a path that is processed in the same way the data was originally collected.

Unlike FINDSDATA, WORDFIX fixes all occurrences of the search argument that it finds in a record. Therefore, it is not necessary to step through the record.

For examples, use PMF to look at TSPs BTNTCEF2 and BTNTCST1 in your base panel data set.

The TSX control lines ADDSDATA (described in "ADDSDATA" on page 71) and DELSDATA (described in "DELSDATA" on page 82) can be used to provide WORDFIX-like functions in a TSX.

Return and Reason Codes

After a WORDFIX control line is run, the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields are set to indicate what happened. These codes are listed in Table 36.

Table 36. WORDFIX Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0		Successful completion.
4	4	The current record did not have a match for the specified argument. No data was changed. Use the VIEW INTERNALS command to see if what you are trying to locate is in the record. If you are using a p-word search, make sure the argument is complete (prefix/data).

Table 36. WORDFIX Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
4	16	An attempt to change some p-word-related data (the data, the prefix, or both) was made, but the assisted-entry panel from which the data was to be collected could not be found or loaded. The WORDFIX was performed, but the validation for that panel was not carried out. Verify that the session member includes the name of the read panel data set where the panel resides. If this panel is supposed to exist but does not, report this to the person at your location who is in charge of panels.
8	4	The requested function was not performed. Your TSP tried to create a data item that was longer than the maximum length accepted by Tivoli Information Management for z/OS. The maximum length for collected data from a response (the s-word, prefixes, and all data associated with those prefixes and control information in the entry) is 256 characters.
8	8	There was not enough storage to process the WORDFIX control line. Contact your system administrator to increase your region size.
8	12	Your response did not meet the validation criteria for the assisted-entry panel. Use PMF to view the assisted-entry panel or check the dictionary for the validation pattern. Refer to the *Tivoli Information Management for z/OS Panel Modification Facility Guide* for more information.
8	16	The WORDFIX control line specified that variable data was to be used (**Get variable data** was set to YES), but the length of the data was zero. You must set up the variable data area and its length before running a TSP with a WORDFIX control line that has YES in the **Get variable data** field. You can call a user exit routine that sets the variable data area length and moves data into it, or use the MOVEVAR control line. Check your TSP to make sure a USEREXIT or MOVEVAR control line appears in the processing path before the WORDFIX control line that caused the unexpected return code. If a USEREXIT control line is in the TSP, check the user exit routine's code to make sure it sets the length field properly.
8	20	An internal logic error occurred in a DBCS function. Contact your Tivoli representative.
8	24	The user data is not a valid mixed string. Check the data and make the changes required to ensure you specify a valid mixed data string.
8	28	The requested data would not be valid Tivoli Information Management for z/OS data. Either no validation data exists in the p-word, or you set **Replace previous reply** to YES but did not specify a prefix or s-word.

Table 36. WORDFIX Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	32	A prefix was specified as new data, but the located p-word had no prefix.

You can use the TESTFIELD control line to test for these return and reason codes. See "TESTFIELD" on page 175 for how to do this.

TSCA Field Usage

Tivoli Information Management for z/OS sets the following TSCA fields after a WORDFIX control line is run. For more information about these fields, see "Terminal Simulator Communications Fields" on page 289.

TSCAFRET
Function return code

TSCAFRES
Function reason code.

If you enter **YES** in the **Get variable data** field and you set the variable data with a user exit routine, the user exit routine must set the length of the variable data in the following TSCA field:

TSCAVDAL
Current user variable data length

If you set the variable data with the MOVEVAR control line, the MOVEVAR control line sets field TSCAVDAL for you.

WRITESOCKET

This control line sends data over a previously established TCP/IP connection. This is a non-blocking operation, so that control is returned to the caller immediately after the data has been given to TCP/IP. If a callable service does not complete successfully, the name of the service called is returned in the NETFUNC REXX variable, the return code is returned in the NETRETC REXX variable, and the reason code is returned in the NETREAC REXX variable. The user should refer to the OS/390 UNIX System Services Messages and Codes manual and the OS/390 UNIX System Services Assembler Callable Services manual for a description of the returned values.

This control line can be called only from a TSX. An ISPF or TSO environment is not required in order to use WRITESOCKET.

The WRITESOCKET Control Line

The format of the WRITEOCKET control line is:

```
CALL BLGTSX 'WRITESOCKET',socketid,datalen,data
```

Parameter Descriptions

1. socketid

Valid reply
The socket identification for this TCP/IP connection. This value was initially returned from the OPENSOCKET control line in the NETSOCKET REXX variable.

Default
None

Required

2. datalen

Valid reply
The length of the data to be sent. The value specified must be greater than 0.

Default
None

Required

3. data

Valid reply
The string data to be sent.

Default
None

Required

Usage Notes and Examples

This is an example of using a WRITESOCKET control line to send data to a waiting server.

Note: Six REXX variables have been defined to return information from the WRITESOCKET control line. These variables (NETSOCKET, NETDATA, NETBYTECOUNT, NETFUNC, NETRETC, and NETREAC) are reset during the processing of WRITESOCKET. It is the responsibility of the TSX to save any data needed for processing.

```
CALL BLGTSX 'WRITESOCKET',SaveSocket,'21','The string to be sent'
```

Return and Reason Codes

After the WRITESOCKET control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 37.

Table 37. WRITESOCKET Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion.
4	4	Not all of the data has been sent. The NETBYTECOUNT REXX variable contains the count of the number of bytes sent. The remaining data bytes must be resent.

Table 37. WRITESOCKET Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	4	The TCP/IP service did not complete successfully. Refer to the OS/390 UNIX System Services Messages and Codes manual and the OS/390 UNIX System Services Assembler Callable Services manual for a description of the following return codes. These REXX variables contain the diagnostic information: **NETFUNC** The name of the Assembler Callable Service being invoked. **NETRETC** The value of the Return_code parameter returned by the service. **NETREAC** The value of the Reason_code parameter returned by the service.

5

Remote Data Resource Terminal Simulator Control Lines

A *remote data resource* is an area in the BLX-SP identified by a unique name. Each resource contains a number of character strings, limited only by the amount of storage available to the BLX-SP address space. This chapter describes some of the terminal simulator control lines that provide functions specific to remote data resources. The *Tivoli Information Management for z/OS Program Administration Guide and Reference* contains some additional information about remote data resources.

Note: The remote data resource control lines listed in this chapter are supported only by TSXs; they are not supported by TSPs.

CLOSERRES

The CLOSERRES control line prevents any more items from being added to a resource.

The state of the specified resource is modified such that new items cannot be added. If the FLUSH option is specified, all existing items are discarded. If the DRAIN option is specified, existing items are not discarded; any existing items can be removed by a TSX via GETRDATA.

This control line can be called only from a TSX.

The CLOSERRES Control Line

The format of the CLOSERRES control line is:

```
CALL BLGTSX 'CLOSERRES','resname',options
```

Parameter Description

1. resname

Valid reply

The name of a remote data resource. It can be from one to eight characters in length, and must contain only alphabetic or numeric characters. It identifies which remote data resource is to be acted upon. The value specified must be enclosed in single quotation marks.

Default

None

Required

2. options

Valid reply

Supported options are:

FLUSH|DRAIN

- **FLUSH** indicates that all existing items are to be discarded.
- **DRAIN** indicates that existing items are not discarded and can be removed by using the GETRDATA TSX.

If neither of these options is specified, the default is **FLUSH**.

Default

FLUSH

Optional

Usage Notes and Examples

This is an example of using a CLOSERRES control line to close a resource and prevent any other items from being added.

```
CALL BLGTSX 'CLOSERRES','resource1','FLUSH'
```

Return and Reason Codes

After the CLOSERRES control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 38.

Table 38. CLOSERRES Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Function completed normally.
4	8	The resource is either already in the process of closing or is already closed.
4	12	The resource does not exist.

GETRDATA

The GETRDATA control line is used to remove an item from a resource. If the specified resource is not defined, it is created with the default limit values. An ENQ for the specified resource name is obtained. An item is retrieved from the resource. If none are available and the WAIT option is specified, execution is suspended until either one becomes available, control of the resource is released, or the resource is closed. Retrieved data is placed in the TSCAVDA variable. The ENQ is released. If the attempt to obtain the ENQ fails, no data is retrieved. If the WAIT option is specified, ownership of the resource is obtained, and held until either the TSX terminates or the resource is released or closed.

The QNAME for the ENQ is BLGRDRS and the RNAME consists of a string containing the subsystem name for the target BLX-SP and the resource name.

This control line can be called only from a TSX.

The GETRDATA Control Line

The format of the GETRDATA control line is:

```
CALL BLGTSX 'GETRDATA','resname',options
```

Parameter Description

1. resname

Valid reply

The name of a remote data resource. It can be from one to eight characters in length, and must contain only alphabetic or numeric characters. It identifies which remote data resource is to be acted upon. The value specified must be enclosed in single quotation marks.

Default

None

Required

2. options

Valid reply

Supported options are:

seconds|WAIT

- **seconds** indicates the number of seconds to wait for data.
- **WAIT** indicates that the TSX is to be suspended until an item can be returned. A specification of WAIT requires ownership of the resource. If the resource is unowned, ownership is assigned to the current TSX. If another TSX issues a GETTRES with WAIT, it will fail.

Default

None

Optional

However, if no value, neither seconds nor WAIT, is specified for the options, data is returned if available, but execution is not suspended.

Usage Notes and Examples

This is an example of using a GETRDATA control line to remove data from a resource.

```
CALL BLGTSX 'GETRDATA','resource1','WAIT'
```

Return and Reason Codes

After the GETRDATA control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 39.

Table 39. GETRDATA Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Function completed normally; TSCAVDA variable contains item.
0	4	The number of seconds specified has elapsed but there is no data to return.
4	4	The resource is empty.
4	8	The resource is empty and items cannot be added.
4	12	Could not obtain ENQ.
4	16	The resource is already owned by another TSX.

OPENRRES

The OPENRRES TSX control line is used to create a controlled resource. It enables a TSX to specify the resource limits at the time the resource is created.

This control line can be called only from a TSX.

The OPENRRES Control Line

The format of the OPENRRES control line is:

```
CALL BLGTSX 'OPENRRES','resname',resetcount,warncount,maxcount
```

Parameter Description

1. resname

Valid reply

The name of a remote data resource. It can be from one to eight characters in length, and must contain only alphabetic or numeric characters. It identifies which remote data resource is to be acted upon. The value specified must be enclosed in single quotation marks.

Default

None

Required

2. resetcount

Valid reply

A numeric value that indicates the number of items in the resource that will enable new items to be added once the maximum has been reached. The resetcount must be greater than zero and less than the value specified for maxcount. The value specified cannot exceed 999 999 999. If resetcount is specified, values must also be specified for warncount and maxcount.

Default

512

Optional

3. warncount

Valid reply

A numeric value that indicates that a warning message is to be issued when the resource contains the specified number of items. Once a warning message has been issued, another message will not be issued until the current item count first reaches the resetcount, then returns to the warning item count. The warning item count must be either zero (in which case a message is never issued) or greater than or equal to the resetcount and less than or equal to the maxcount. The value specified may not exceed 999 999 999. If warncount is specified, values must also be specified for resetcount and maxcount.

Default

768

Optional

4. maxcount

Valid reply

A numeric value that indicates the maximum number of items that can be added to a resource. Once the current item count reaches the maximum allowed item count, new items cannot be added until the current item count returns to the resetcount. The value specified for maxcount must be greater than the value specified for resetcount and must be equal to or greater than the warncount. The value specified may not exceed 999 999 999. If maxcount is specified, values must also be specified for resetcount and warncount.

Default

1024

Optional

Usage Notes and Examples

This is an example of using an OPENRRES control line to create a controlled resource.

```
CALL BLGTSX 'OPENRRES','resource1',1000,2000,3000
```

Return and Reason Codes

After the OPENRRES control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 40.

Table 40. OPENRRES Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Function completed normally.
4	8	Invalid resource limit values specified.
4	12	Could not obtain ENQ for resource.
4	16	Resource already owned by another TSX.

PUTRDATA

The PUTRDATA control line is used to add an item to a resource. The specified data string is added to the specified resource.

This control line can be called only from a TSX.

The PUTRDATA Control Line

The format of the PUTRDATA control line is:

```
CALL BLGTSX 'PUTRDATA','resname',data
```

Parameter Description

1. resname

Valid reply

The name of a remote data resource. It can be from one to eight characters in length, and must contain only alphabetic or numeric characters. It identifies which remote data resource is to be acted upon. The value specified must be enclosed in single quotation marks.

Default
None

Required

2. data

Valid reply
A character string not greater than 512 characters.

Default
None

Required

Usage Notes and Examples

This example uses a PUTRDATA control line to put data into a resource.

```
CALL BLGTSX 'PUTRDATA','resource1',data
```

Return and Reason Codes

After the PUTRDATA control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 41.

Table 41. PUTRDATA Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Function completed normally; item was added to resource.
4	8	Resource is not accepting new items.
4	12	The resource does not exist.

QUERYRRES

The QUERYRRES control line is used to obtain information about a resource. The following information is returned in the TSCAVDA variable:

- Resource ownership

  UNOWNED
  The specified resource is not controlled by any TSX.

  OWNED
  The specified resource is controlled by another TSX.

  OWNER
  The specified resource is controlled by the current TSX.

- Resource state

  OPENED
  Items can be added or removed from the specified resource.

  CLOSED
  The resource exists and is empty; no data can be added to it.

DRAINING
Items can only be removed from the resource.

RELEASED
Items are not being removed from the resource.

UNDEFINED
The resource does not exist.

- Current item count
- Total number of items processed since the resource was created.
- Limits for resetcount, warncount, and maxcount.

An example of the contents of the TSCAVDA variable is:

```
UNOWNED OPENED 10 20 10 20 30
```

This control line can be called only from a TSX.

The QUERYRRES Control Line

The format of the QUERYRRES control line is:

```
CALL BLGTSX 'QUERYRRES','resname'
```

Parameter Description

1. resname

Valid reply
The name of a remote data resource. It can be from one to eight characters in length, and must contain only alphabetic or numeric characters. It identifies which remote data resource is to be acted upon. The value specified must be enclosed in single quotation marks.

Default
None

Required

Usage Notes and Examples

This example uses a QUERYRRES control line to obtain information about a resource.

```
CALL BLGTSX 'QUERYRRES','resource1'
```

RELEASERRES

The RELEASERRES control line is used to release control of a resource. Control of a specified resource is released, and is left in a state in which new items can be added until the maximum allowed item count is reached.

This control line can be called only from a TSX.

The RELEASERRES Control Line

The format of the RELEASERRES control line is:

```
CALL BLGTSX 'RELEASERRES','resname'
```

Parameter Description

1. resname

Valid reply

The name of a remote data resource. It can be from one to eight characters in length, and must contain only alphabetic or numeric characters. It identifies which remote data resource is to be acted upon. The value specified must be enclosed in single quotation marks.

Default

None

Required

Usage Notes and Examples

This example uses a RELEASERRES control line to release control of a resource.

```
CALL BLGTSX 'RELEASERRES','resource1'
```

Return and Reason Codes

After the RELEASERRES control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 42.

Table 42. RELEASERRES Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Function completed normally; resource was released.
4	8	Resource is in the process of being released.
4	12	The resource does not exist.

SETRRES

The SETRRES control line is used to set limits for a resource.

This control line can be called only from a TSX.

The SETRRES Control Line

The format of the SETRRES control line is:

```
CALL BLGTSX 'SETRRES','resname',resetcount,warncount,maxcount
```

Parameter Description

1. resname

Valid reply

The name of a remote data resource. It can be from one to eight characters in length, and must contain only alphabetic or numeric characters. It identifies which remote data resource is to be acted upon. The value specified must be enclosed in single quotation marks.

Default

None

Required

2. resetcount

Valid reply

A numeric value that indicates the number of items in the resource that will enable new items to be added once the maximum has been reached. The resetcount must be greater than zero and less than the value specified for maxcount. The value specified cannot exceed 999 999 999. If resetcount is specified, values must also be specified for warncount and maxcount.

Default

512

Optional

3. warncount

Valid reply

A numeric value that indicates that a warning message is to be issued when the resource contains the specified number of items. Once a warning message has been issued, another message will not be issued until the current item count first reaches the resetcount, then returns to the warning item count. The warning item count must be either zero (in which case a message is never issued) or greater than or equal to the resetcount and less than or equal to the maxcount. The value specified may not exceed 999 999 999. If warncount is specified, values must also be specified for resetcount and maxcount.

Default

768

Optional

4. maxcount

Valid reply

A numeric value that indicates the maximum number of items that can be added to a resource. Once the current item count reaches the maximum allowed item count, new items cannot be added until the current item count returns to the resetcount. The value specified for maxcount must be greater than the value specified for resetcount and must be equal to or greater than the warncount. The value specified may not exceed 999 999 999. If maxcount is specified, values must also be specified for resetcount and warncount.

Default

1024

Optional

Usage Notes and Examples

This example uses a SETRRES control line to set limits for a controlled resource.

```
CALL BLGTSX 'SETRRES','resource1',1000,2000,3000
```

Return and Reason Codes

After the SETRRRES control line is run, Tivoli Information Management for z/OS sets the TSCA return code (TSCAFRET) and reason code (TSCAFRES) fields to indicate what happened. These codes are listed in Table 43 on page 250.

Table 43. SETRRES Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Function completed normally.
4	8	Invalid resource limit values specified.
4	12	Resource not found.

6

Testing Terminal Simulator Panels (TSPs) and EXECS (TSXs)

This chapter discusses the tools you can use to test the accuracy of TSPs and TSXs. Tivoli Information Management for z/OS does not check for correct panels or field values; it only supplies you with the tools to do so. The tools discussed here are four of the control lines described in Creating Terminal Simulator Control Lines: PRINT, TESTFIELD, TESTFLOW, and TRACE. Only PRINT and TRACE are applicable to a TSX. The functions of the TSP TESTFIELD and TESTFLOW control lines can be done in a TSX using conditional program flow controls (if, do, while, etc.) available in the REXX programming language.

When you create TSPs and TSXs, you must keep in mind that some control lines, especially WORDFIX, FLATTEN, UNFLATTEN, ADDSDATA, and DELSDATA can do considerable damage to your database if they are used incorrectly.

Because of the flexibility of Tivoli Information Management for z/OS, it is important that you incorporate built-in testing in your TSP or TSX. If a panel is removed or data, such as a selection number, is changed, your TSP or TSX should perform certain tests to make sure that those changes do not affect the TSP or TSX. A TSX could perform a similar test by checking the panel name using TSCA field TSCACPNL.

Using the PRINT Control Line

The PRINT control line is valid in a TSP and a TSX, and it enables you to print current and saved messages, the current panel, and the TSCA so that you can check the field values. If your TSP or TSX branches to an error routine, you can find out why by including the PRINT control line.

For example, in a TSP, assume that data was put in the command line reply buffer by an ADDDATA control line. When the PROCESS control line runs, you do not get the results you expected. You can set up the following error handling routine to take control:

```
TESTFLOW   ADD     (branch if correct panel)
PRINT              (TSCA, messages, panel)
LABEL      ADD
RETURN
```

Similarly, in a TSX, if you do not get the results you expected after a PROCESS control line (the TSCAFRET and TSCAFRES REXX equivalent variables are not both 0), your REXX code can invoke the PRINT control line:

```
IF TSCAFRETˆ=0 | TSCAFRETˆ=0 THEN,
     CALL BLGTSX 'PRINT','MESSAGES','SCREEN','TSCA';
```

Using the TESTFIELD Control Line

The TSP TESTFIELD control line enables you to test for expected character or numeric values in any TSCA field. You can test for the return and reason codes that most control lines set. For example, assume that a FINDSDATA control line is processed. Before you do anything with the data that was returned from that control line, you enter a TESTFIELD control line to test the return code. If TESTFIELD finds a return code of zero, the next control line is processed. If the code is not zero, the FINDSDATA did not work as expected and a branch is taken to the error label specified on the TSP TESTFIELD control line.

Even after a TSP is in production, continue to use this control line to check critical field values, because system changes can affect the processing result of the TSP.

Refer to “Field Checking in a TSX” for information on how to handle field checking in a TSX.

Field Checking in a TSX

You can perform field checking within a TSX directly using the standard program flow controls available in the REXX programming language. For example, assume that a FINDSDATA control line is processed. Before you do anything with the data that was returned from that control line, you should check the TSCA REXX equivalent variable TSCASDF. The TSCASDF contains the data found by FINDSDATA. If the length of the data is zero, then FINDSDATA did not locate the desired data. Example REXX code:

```
CALL BLGTSX 'FINDSDATA','PERA/.',,'LAST';
IF LENGTH(TSCASDF)=0 THEN,
   DO;
      MSGTXT='PLEASE ENTER AN ASSIGNEE NAME.'
      CALL BLGTSX 'MESSAGE',,MSGTXT,'SAVE';
   END;
```

Of course, you also have the option of testing the values of the TSCAFRET and TSCAFRES REXX equivalent variables to determine the success or failure of a TSX control line. Example REXX code:

```
SWORD='S0E01';                    /* Description text */
CALL BLGTSX 'READDICT',SWORD;     /* Get s-word       */
CALL BLGTSX 'FINDTEXT',TSCARSD;   /* Get text         */
IF TSCAFRET=0 & TSCAFRES=0 THEN,  /* Successful?      */
   DO;
      MSGTXT='Record text successfully retrieved.';
      CALL BLGTSX 'MESSAGE',,MSGTXT,'SAVE';
   END;
```

Using the TESTFLOW Control Line

The TSP TESTFLOW control line enables you to check for panel flow. This control line is similar to TESTFIELD, but instead of testing for field values, it tests to see whether the current panel is the expected one.

When you use ADDDATA and PROCESS control lines in a TSP, you expect to reach a certain panel. When you use Tivoli Information Management for z/OS interactively, that panel appears on your screen. When processing an IRC from within a TSP, especially in batch mode, you do not know if you have reached the correct panel. The panel flow might have changed since the TSP was created.

To test for the correct panel, you can use a TESTFLOW control line before you add any information to the panel. Or, instead of using TESTFLOW, you can include a user exit routine that tests for the current panel, which is identified in the TSCA field TSCACPNL.

You can also use TESTFLOW to test for the presence of messages. For example, when your TSP files a record, you might check to be sure you get the *Record successfully filed* message.

As with TESTFIELD, it is recommended that you continue to use this control line even after a TSP is in production, so that the panel flow and messages are always checked.

Refer to "Panel Checking in a TSX" and "Message Checking in a TSX" for information on how to handle panel and message checking in a TSX.

Panel Checking in a TSX

You can test the current panel within a TSX directly using conditional program flow controls (if, do, while, etc.) available in the REXX programming language. Before a TSX chooses a selection on a data entry panel, the TSX should confirm the current panel. For example, before entering in an assignee name into a problem record, the TSX should verify that the current panel is BLG0B200. Example REXX code:

```
IF TSCACPNL='BLG0B200' THEN,
   DO;
      REPLY='1,DOE/JOHN';
      CALL BLGTSX 'PROCESS',REPLY;
   END;
ELSE
   EXIT;
```

Message Checking in a TSX

If a TSX control line generates messages, the message numbers along with their text will be stored in the REXX compound variable BLG_MESSAGE. and the total number of messages will be stored in BLG_MESSAGE.0. To scan for a particular message, refer to the example code below:

```
found=0
do i=1 to BLG_MESSAGE.0 while (found=0)
   if substr(BLG_MESSAGE.I,1,8)='BLG03058' then do
      /* Code to do anything you want to do here */
      found=1
   end
end
```

Since a TSX has access to the message text (a TSP only has access to the message number), a TSX can extract data from a message such as the RNID of the record that just filed successfully.

Syntax Checking in a TSX

If a call to a TSX control line has a parameter error (for example, a required parameter is not specified), it will cause a REXX syntax error. Messages describing the syntax error are stored in the REXX compound variable BLG_ERROR. The total number of syntax errors is stored in BLG_ERROR.0. Normally, an EXEC will end when it has a syntax error, and you would never get a chance to see what's in BLG_ERROR. In order to see what's there, you can do the following:

- At the top of the TSX, code the following:

```
signal on syntax name synrtn
parse source . . execname .         /* Get exec name              */
```

  This tells REXX to call a subroutine named 'synrtn' if any syntax errors occur. The name of the TSX is also saved for use by the syntax routine.

- Somewhere in your TSX, include the following subroutine which will print out the line number in error with the name of the TSX that had the syntax error along with the messages issued by Tivoli Information Management for z/OS.

```
synrtn:
say 'Syntax error at line' sigl 'in TSX' execname /* Print location*/
 if BLG_ERROR.0/='BLG_ERROR.0' then /* Have control line errors?   */
  do i = 1 to BLG_ERROR.0           /* Loop through the messages   */
   say '>'BLG_ERROR.i'<'            /* Display error message       */
  end
exit
```

Refer to the member BLGFLAT in the SBLMTSX data set to see an example of a TSX that uses this technique to handle syntax errors.

Using the TSP TRACE Control Line

For a TSP, TRACE produces a report that shows you how the TSP was processed. It indicates which control lines were processed, the return and reason codes for those control lines, the order in which the control lines were processed, which branches were taken, and which other TSPs or TSXs were linked to for processing. In other words, TRACE lets you analyze the flow of a TSP and evaluate the results of an operation.

To see the complete flow of the TSP, enter TRACE as your first control line. If the beginning of the TSP only builds a command line reply buffer and you don't want to follow that flow, enter the TRACE control line just before you process the reply buffer. If the TSP branches to an error routine, you will know that you did not build the command line reply buffer correctly.

You probably will want to remove the TRACE control line after you determine that the TSP's flow is correct, especially if you continue to use TESTFLOW and TESTFIELD to check for correct panels and field values.

An example of the TRACE output is shown in Figure 6 on page 255. This example shows the flow of the TSP created in Chapter 2, "Creating a Terminal Simulator Panel Flow" on page 9.

Using the TRACE Command

If you prefer, you can use the TRACE command instead of the TRACE control line. This command provides the same capability as the TRACE control line, except that you specify the trace interactively on the command line before you run your TSP or TSX. In this case, you do not need to update your TSP to include a TRACE control line. Each TSP that is processed will be traced until you use the TRACE command to turn tracing back off.

Note: It is the responsibility of your TSX to check the value of BLGTRACE and specify the desired REXX TRACE parameters you prefer. Also, TSX REXX TRACE

information does not go to the output destination specified in the BLGTRACE DD statement. REXX controls the output destination, and typically the destination is the terminal.

For more information on the TRACE command, refer to the *Tivoli Information Management for z/OS User's Guide* .

Running traces on multiple sessions concurrently can produce unexpected results. This applies to the TRACE command as well as the trace control line. The completeness of the trace depends upon the output destination specified in the BLGTRACE DD statement. If the output destination is a data set, trace information for some sessions might not be recorded. If the output destination is a SYSOUT device, trace information for all sessions is recorded completely.

```
LN  FUNCTION  PANEL    LITERAL/          TSCA FLAGS    S-WORD      PREFIX      FIELD NEW DATA, S-WORD LABEL/   ARGUMENT   FUNCTION
#     NAME    NAME      USER DATA        0  1  2  3   INDEX WORD  INDEX WORD  NAME      OR P-WORD    TARGET   PANEL/MSG  EXIT

      ****TSCAFRET=0 TSCAFRES=0****
002 TRACE     BLGTSP01                  04 00 00 00                                                            BLG01312
      ****TSCAFRET=0 TSCAFRES=0****
003 ADDDATA   BLGTSP01 3,2,6,1,SE + PER 00 00 00 00                                                            BLG01300
      ****TSCAFRET=0 TSCAFRES=0****
004 PROCESS   BLGTSP01                  00 00 00 00                                 ERROR                      BLG01303
      ****TSCAFRET=0 TSCAFRES=0****
005 TESTFLOW  BLGTSP01                  10 00 00 00                                 DONE     BLG19214         BLG01302
      ****TSCAFRET=0 TSCAFRES=0****
006 TESTFIELD BLGTSP01 1                00 00 00 00                                 JUSTONE                    BLG01301
      ****TSCAFRET=0 TSCAFRES=0****
007 ADDDATA   BLGTSP01 LINECMD UU,DOWN  00 00 00 00                                                            BLG01300
      ****TSCAFRET=0 TSCAFRES=0****
008 PROCESS   BLGTSP01                  00 00 00 00                                 ERROR                      BLG01303
      ****TSCAFRET=0 TSCAFRES=4****
009 LABEL     BLGTSP01                  00 00 00 00                                 UPDATE
      ****TSCAFRET=0 TSCAFRES=4****
010 ADDDATA   BLGTSP01 2,1,JONES,,9     00 00 00 00                                                            BLG01300
      ****TSCAFRET=0 TSCAFRES=0****
011 PROCESS   BLGTSP01                  00 00 00 00                                 ERROR                      BLG01303
      ****TSCAFRET=0 TSCAFRES=0****
012 TESTFLOW  BLGTSP01                  00 00 00 00                                 UPDATE   BLG0BUOO          BLG01302
      ****TSCAFRET=0 TSCAFRES=0****
009 LABEL     BLGTSP01                  00 00 00 00                                 UPDATE
      ****TSCAFRET=0 TSCAFRES=0****
010 ADDDATA   BLGTSP01 2,1,JONES,,9     00 00 00 00                                                            BLG01300
      ****TSCAFRET=0 TSCAFRES=0****
011 PROCESS   BLGTSP01                  00 00 00 00                                 ERROR                      BLG01303
      ****TSCAFRET=0 TSCAFRES=0****
012 TESTFLOW  BLGTSP01                  00 00 00 00                                 UPDATE   BLG0BUOO          BLG01302
      ****TSCAFRET=0 TSCAFRES=0****
013 BRANCH    BLGTSP01                  00 00 00 00                                 DONE                       BLG01304
      ****TSCAFRET=0 TSCAFRES=0****
017 LABEL     BLGTSP01                  00 00 00 00                                 DONE
      ****TSCAFRET=0 TSCAFRES=0****
018 RETURN    BLGTSP01                  00 00 00 00                                                            BLG01309
```

Figure 6. Example of TRACE Output

Column Description

LN# The line number of the TSP control line being processed

Function Name
The function specified on the current control line

Panel Name
The panel name of the current TSP

Literal/User Data
Literal or user data entered for the control line being processed (a maximum of 15 bytes of the data appears on each line of the report)

TSCA Flag Fields
The TSCA fields, TSCA0FLG, TSCA1FLG, TSCA2FLG, and TSCA3FLG in hexadecimal format

S-Word Index Word
The s-word index and s-word entered for the control line being processed

Prefix index Word
The prefix index and p-word entered for the control line being processed

Field Name
The name of the TSCA field entered on a TESTFIELD control line as the field to test

New Data, S-Word, or P-Word
The replacement data for a WORDFIX control line (a maximum of 15 bytes of the data appears on each line of the report)

Label/Target
The label name in a LABEL control line, or the target label of a BRANCH control line

Argument Panel/Msg
The panel name or message identifier entered as an argument for a control line

Function Exit
The routine name specified as an exit routine or a TSP function exit.

TRACE TSX Considerations

In a TSX, variable BLGTRACE will be set to 1 when you use the TRACE command to activate TSP and TSX tracing. BLGTRACE will also be set to 1 when you specify **YES** for Trace LINK function on the TSP TRACE control line. It is the responsibility of your REXX code to test the value stored in the BLGTRACE variable and to issue the corresponding desired REXX TRACE statement. Example REXX code:

```
IF BLGTRACE=1 THEN, /* TSP/TSX Tracing Active?              */
  TRACE RESULTS     /* Start REXX trace with desired options */
```

This keeps you from having to update your TSXs with trace statements. If you specify those statements at the beginning of your TSX, the entire TSX will be traced. You might choose to specify those statements at other locations in the TSX or even to turn REXX tracing off.

Note: TSX REXX TRACE information does not go to the output destination specified in the BLGTRACE DD statement. REXX controls the output destination, and typically the destination is the terminal.

7

Running Terminal Simulator Panels (TSPs) and Terminal Simulator Execs (TSXs)

You can run Terminal Simulator Programs (TSPs) in batch mode, at product invocation, while you are interactively using Tivoli Information Management for z/OS, or from other TSPs. The same is true for terminal simulator EXECs (TSXs); you can run a TSX anywhere that you can run a TSP. You can even use the LINK control line to call a TSX from a TSP or to call a TSP from a TSX. This chapter describes the procedures for starting a TSP or a TSX.

Note: Before you try to run a TSX, make sure that you have allocated the DD BLGTSX to the library that contains your TSX REXX EXECs.

Running a TSP or a TSX from the Command Line

You can run a TSP or a TSX from the command line using the following techniques:

- With the RUN command
- With a command alias in the COMMAND record

Using the RUN Command to Run a TSP or a TSX

Issue the RUN command with an operand specifying which TSP or TSX you want to run. The operand can be a tspname, tsxname, or aliasname. If the name or alias is for a TSX, you can also enter additional operands that will be passed as an argument to the TSX. This shows the syntax of the RUN command.

```
RUN tspname
RUN tsxname
RUN tsxname argument
RUN aliasname
RUN aliasname argument                /* If aliasname is for a TSX */
```

The *Tivoli Information Management for z/OS User's Guide* contains additional information about the RUN command.

Using a Command Alias to Run a TSP or a TSX

A command alias can be used to make it easier to run a TSP or a TSX from the command line.

In order to define a command alias to run a TSP or a TSX, add an entry to the COMMAND record with the desired **Command Alias** name and the RUN command (with any operands) as the **Actual Command String**. This example contains several entries defining a command alias for the RUN command.

```
Command Alias  Actual Command String
OPEN           RUN OPENPROB            /* TSP OPENPROB is run */
CLOSE          RUN CLSPTSX             /* TSX CLSPTSX is run  */
CLSLATE        RUN CLSPTSX LATE        /* TSX CLSPTSX is run.
                                          Argument LATE is passed.  */
OC             RUN OC                  /* The TSP or TSX associated
                                          with aliasname OC is run. */
```

To use a command alias, simply enter it as you would a command. Operands specified with the command alias are added to the end of the **Actual Command String**. For example, if you entered:

```
CLSLATE 4
```

the actual command entered on the command line would be:

```
RUN CLSPTSX LATE 4
```

The string LATE 4 will be passed to TSX CLSPTSX as a single argument which the TSX can parse as needed.

The *Tivoli Information Management for z/OS Program Administration Guide and Reference* contains additional information about setting up a **command alias** in the COMMAND record.

Running a TSP or a TSX at Product Invocation

You can run a TSP or a TSX at product invocation using the ISPSTART statement. The ISPSTART statement accepts a TSP parameter, an IRC parameter, or an SRC parameter. If you specify the IRC or SRC parameters on the ISPSTART statement along with the TSP parameter, the TSP parameter is processed first after the product proprietary panel appears and the user returns from it. These are some examples of the ISPSTART statement syntax for the TSP parameter.

```
ISPSTART PGM(BLGINIT) PARM(TSP(tspname))
or                    PARM(TSP(tsxname))
or                    PARM(TSP(aliasname))
```

You can use the IRC parameter to run a TSP or a TSX using the RUN command or a command alias. These are some examples of the ISPSTART statement syntax for the IRC parameter.

```
ISPSTART PGM(BLGINIT) PARM(IRC(RUN tspname))
or                    PARM(IRC(RUN tsxname))
or                    PARM(IRC(RUN tsxname argument))
or                    PARM(IRC(RUN aliasname))
or                    PARM(IRC(commandalias))
```

Refer to the *Tivoli Information Management for z/OS Planning and Installation Guide and Reference* for more information on the ISPSTART statement and how it is used to invoke Tivoli Information Management for z/OS.

Running TSPs or TSXs in a Batch Environment

You run a TSP or a TSX in batch mode the same way that you run a TSP or a TSX at product invocation using the ISPSTART statement. The examples contained in “Running a TSP or a TSX at Product Invocation” show how the ISPSTART statement can be used to run a TSP or a TSX in a batch environment.

If you specify the IRC or SRC parameters on the ISPSTART statement along with the TSP parameter, the TSP parameter is processed first.

Note: In order for a Tivoli Information Management for z/OS batch job to terminate cleanly (RC=X'0'), the batch job must end due to the Tivoli Information Management for z/OS QUIT command.

Refer to the *Tivoli Information Management for z/OS Planning and Installation Guide and Reference* for more information on the ISPSTART statement and how it is used to invoke Tivoli Information Management for z/OS in batch mode.

Running a TSP or a TSX from a Control Panel

The 002B Function Code

You can also call a TSP or a TSX from a control panel during an interactive or batch session. At the point in the control panel where you want the TSP or TSX initialized, specify the following on a FLOW control line:

- The 002B function code
- The name or alias of the TSP or TSX you want to run
- Optionally, a target panel for the control panel

The target panel is the panel from which the TSP or TSX actually begins to run. If you do not specify a target panel, the TSP or TSX will run on the next displayable panel. (This is the same as entering 000B on a control line, except that a TSP or TSX is started rather than a program exit.)

For example, after you enter the priority for a problem record, you could run a TSP or a TSX that calculates a fixed target date and then automatically enters the date into the record.

You will need to make these changes:

1. Update panel BLG6CPRI, the assisted-entry panel for current priority:

```
Create target    = The control panel that calls the TSP or TSX
Inquiry target   = BLG0E201, or the name of the panel from which
                   BLG6CPRI was called
Return to caller = NO
```

2. Create a control panel with the name specified as the create target for panel BLG6CPRI:

```
Control line type     = FLOW
Function code index   = 002B
True target panel     = BLG0B100, or the name of the panel from which
                        BLG6CPRI was called
Program exit/TSP name = name or alias of the TSP or TSX to run
```

The TSP or TSX is initialized at the time the FLOW control line is run, but the TSP or TSX is not actually started until the panel flow reaches an interactive (displayable) panel. If the 002B control line contains a true target, flow passes to that panel; otherwise, processing of the control panel continues until it completes and flow passes to another panel. In either case, if the next panel is an interactive (displayable) panel, the TSP/TSX will run on that panel. However, if the next panel is another control panel, that panel and any subsequent control panels will be processed until the flow eventually reaches an interactive panel, at which time the TSP/TSX will run.

You can determine which panel a TSP/TSX is running on by checking the TSCACPNL field at the start of the TSP/TSX.

If a 002B control line is used within a multiple test begin and end group, subsequent control lines are skipped and processing resumes with the first line after the multiple test ends.

The 001B Function Code

You can also schedule that a TSP or TSX be run when a record is filed by specifying the following on a FLOW control line:

- The 001B function code
- The name or alias of the TSP or TSX you want to run

Typically, a FLOW with a 001B function code would be specified on a record type file time control panel. If the record is not filed, the TSPs and TSXs are never run. If the record is filed, the TSPs and TSXs are initialized and run in last-in, first-out order. The TSP, TSX, and alias names remain connected to the record until it is filed or the entry is canceled.

Note: For both the 001B and the 002B function codes, the FLOW line must be on a control panel that Tivoli Information Management for z/OS processes rather than on a control panel that contains only information that Tivoli Information Management for z/OS uses.

When the TSP or TSX Is Actually Started

A TSP or TSX initialized with the 002B function code is not actually started until the panel flow reaches an interactive panel. For example, suppose you want to call a TSP or a TSX upon leaving panel A on the way to panel B. Also, assume that the control panel (created to call the TSP or TSX) has a FLOW control line with a target of panel B. What you expect to happen is:

```
Panel A----> Control Panel----> TSP or TSX----> Panel B
```

However, if panel B is the type of interactive panel that end users can see, the following is the actual flow:

```
Panel A----> Control Panel----> Panel A----> TSP or TSX----> Panel B
```

Because TSPs and TSXs are actually started after a displayable panel is reached, the TSP or TSX is not started until the panel flow reaches panel B. If the TSP or TSX does not change the panel flow, panel B appears after the TSP or TSX ends. If the TSP or TSX issues commands to change the panel flow, the panel that you see is based on the new flow.

Suppose that you are building a search argument. You make a panel selection that calls a control panel to start a TSP or a TSX and that also performs a function code 0008 (INITIALIZE). In this case, Tivoli Information Management for z/OS clears your search argument and diverts the panel flow to the Primary Options Menu (BLG0EN20) before starting your TSP or TSX.

In another example, suppose that you want a TSP or a TSX to process a record whenever you make a panel selection to file it. In this case, the next panel that accepts a user reply is the panel from which you started the record create or update. By that time, the record has already been filed and is no longer available for the TSP or TSX to process. However, there is a way to do this. The RNID/ of the last record filed is in the TSCA. You can use that to update the record just filed, display the record to get data out of it, or perform a search.

How To Locate a TSP or TSX

A TSP and a TSX can have the same name. So can a TSP and an alias name or a TSX and an alias name. Tivoli Information Management for z/OS uses the following guidelines to determine the TSP, TSX, or aliasname to use:

- The ALIAS record is checked to see if the name is defined.
  - If so, the ″real name″ and type (TSP, TSX or undefined) is obtained.
  - If not, the name is assumed to be a real name and the type is undefined.
- If the type is TSP or undefined, an attempt is made to load the TSP.
  - If no panel is found and the type is defined as TSP, an error message is issued.
  - If no panel is found and the type is undefined, call REXX to load/run the TSX. REXX issues a message if it cannot load the EXEC.

Calling a TSP or TSX from Another TSP or TSX

You can call a TSP or TSX from within another TSP or TSX by using the LINK control line. When the linked-to TSP or TSX completes processing, control is returned to the calling TSP or TSX. If you specify a starting panel name in the Starting panel name field of a TSP panel, Tivoli Information Management for z/OS verifies that the name matches the current panel before initializing the linked-to TSP. If the name entered in the Starting panel name field of the TSP does not match the current panel, Tivoli Information Management for z/OS cannot initialize the TSP and issues an error message. In a TSX, it is the responsibility of the TSX to check the contents of the TSX variable TSCACPNL to verify the starting panel if desired. For more information about the LINK control line, see “LINK” on page 130.

You can indirectly call a TSP from within another TSP when one or more ADDDATA control lines, followed by a PROCESS control line, pass control to a control panel that contains a control line with a 002B or a 001B function code index. Also, you can directly call a TSP from within another TSP by using an ADDDATA control line that contains the RUN command. The same is true for a TSX except that the ADDDATA and PROCESS functions are combined in the PROCESS control line for a TSX.

Be aware that having several TSPs or TSXs running at the same time causes TSCA fields to be shared. Examine the panel flow to verify that sharing these fields does not cause unexpected data to be used by another TSP or TSX that is either linked to by the first one or started by the first one by a response to a panel.

Message Handling during TSP and TSX Processing

Because TSPs and TSXs can run in batch mode and during an interactive session, you must consider both modes for message-handling purposes. Tivoli Information Management for z/OS saves any messages that exist before processing a TSP or TSX. You can display these messages after the processing of the TSP or TSX is complete, unless the TSP or TSX issues the QUIT command. Depending on which mode you use, messages that are pending after a TSP or TSX completes are returned differently. If a TSP or TSX is running during an interactive session, the messages are displayed automatically. If a TSP or TSX is running in batch mode, the messages are sent to the destination specified by the SYSPRINT DD statement.

If Tivoli Information Management for z/OS processes a QUIT command in either mode, it discards all pending messages and ends. If you do not want to lose these messages, include a PRINT control line before the control line that processes the QUIT command.

While the TSP or TSX is running, Tivoli Information Management for z/OS handles message processing in the same manner for each mode. Tivoli Information Management for z/OS generates messages whenever it processes a MESSAGE control line. The PROCESS control line can also generate messages. A null PROCESS control line that occurs when the command-line reply buffer is empty causes the deletion of messages from the current message chain. If you want to record these messages before they are deleted, you must enter a PRINT control line before the PROCESS control line. If you want these messages to be saved in a TSP, you must add them to the saved message chain by specifying YES for the Save existing messages field on the PROCESS control line. If you want these messages to be saved in a TSX, you must specify SAVE for the save parameter on the TSX PROCESS control line.

TSP and TSX Processing for Three Classes of Messages

The three classes of messages are informational, warning, and severe. The following describes the processing associated with each message class:

Informational. Tivoli Information Management for z/OS always treats messages generated using the MESSAGE control line as informational messages. When you set the **Save generated message** field to YES, Tivoli Information Management for z/OS unconditionally saves these messages and restores them to the current message chain when TSP processing completes, regardless of other processing that occurs within the TSP. The same is true when you specify SAVE for the save parameter on the TSX PROCESS control line. Tivoli Information Management for z/OS can also generate informational messages as a result of a PROCESS control line. Informational messages do not affect processing.

Warning. A TSP or TSX will not run if warning messages occur. For both TSP keyword and function code 002B and 001B calls to a TSP or TSX, Tivoli Information Management for z/OS can generate a warning message between the time that the TSP or TSX is initialized and the time that it actually runs. If this occurs, the TSP or TSX ends without processing any control lines. If warning messages are generated in response to a PROCESS control line, Tivoli Information Management for z/OS returns control to the TSP or TSX that contains the control line. Any TSPs or TSXs that were called as a result of the control line end, and processing resumes at the error label for the TSP PROCESS control line or after the TSX PROCESS control line in the REXX code. The TSX must check the TSCAFRES and TSCAFRET to determine what action to take next.

Severe. A TSP or TSX does not run if severe messages exist. If TSP or TSX initialization results in a severe error, Tivoli Information Management for z/OS generates an error message and the TSP or TSX ends. If severe messages are generated as a result of a PROCESS control line, Tivoli Information Management for z/OS returns control to the TSP or TSX that contains the control line. Any TSPs or TSXs that were called as a result of the control line end, and processing resumes at the TSP error label for the control line or after the TSX PROCESS control line is in the REXX code. The TSX must check the values of the TSCAFRET and TSCAFRES to determine what action to take next. In the error routine, evaluate whether the operation is to be retried, or an error message is to be generated and the TSP or TSX ended. The reply buffer is not restored when a severe message is found during processing.

8

User Exits

This chapter describes the user exits supplied with Tivoli Information Management for z/OS. After you understand how these user exits function, you can use them in your TSPs. You can also replace them with user exits of your own design to conform to your own special purposes.

Read "USEREXIT" on page 193 before writing any user exit routines. It gives you hints about writing routines and discusses points to remember when writing routines in PL/I, Assembler, or VS COBOL II.

The user exits in this chapter are presented as follows:

- "Application Program Interface User Exits"
- "Configuration Migration User Exits" on page 270
- "Database Administration User Exits" on page 277
- "Escalation and Notification User Exits" on page 278
- "General-purpose User Exits" on page 279

These user exits require various types of data as input:

- Fields in the TSCA that are expected to be set before this exit runs. The input is described by using the TSCA field name.
- Data that should be entered when the USEREXIT control line is used in a TSP. The field name on the USEREXIT panel is specified.
- The variable data area, which does not have a TSCA name.

Note: User exits which require that information be specified on the TSP USEREXIT control line panels (BLM8CU9P, BLG8CU9Q) cannot be run from a TSX. Refer to the Environment section of each of the user exits in this chapter to determine whether the user exit can be run from a TSX.

Application Program Interface User Exits

Most application program interface (API) user exits can be used only within the API environment. In most cases, if an exit is operating outside the API environment, Tivoli Information Management for z/OS stops with ABEND code 700, reason code 32.

You can use the following exits outside the API environment: BLGEXDEL, BLGJAUTH, and BLGTSAPI.

BLGEXDEL–Delete Unusable Record

CAUTION:
User exit BLGEXDEL deletes the root VSAM key without checking for authority. To protect your database, have the TSP that calls this user exit check for authorization.

User exit BLGEXDEL deletes a record using the root VSAM key, which is passed in the TSCA variable data area.

Even though the record is deleted, it is still cognized. The record might show as *deleted* on a search results list. To prevent this, run the SDIDS build utility (BLGUT1) after calling this user exit.

For an example of how BLGEXDEL can be used, use PMF to look at API TSP BLGAPI10 in your base panel data set. The *Tivoli Information Management for z/OS Application Program Interface Guide* has additional information about this user exit.

This user exit can be used outside the API environment.

Input Root VSAM key (in character format) in the TSCA variable data area.

The database number:

- If the exit is used in the API environment, the database number is taken from field PICADBID.
- If the exit is used outside the API environment, the database number is taken from the user's profile. If the profile does not contain a database number, database 5 is used.

Output
Return and reason codes as listed in Table 44.

Environment
TSP and TSX

Table 44. BLGEXDEL Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion. The record was deleted.
0	4	A storage allocation error occurred. Some records were deleted.
0	8	A database access error occurred. Some records were deleted.
4	4	The variable data length (TSCAVDAL) is not equal to 8.
4	8	The database was not found.
4	12	The database is not read/write.
4	16	Enqueuing the record on the root VSAM key was not successful.
8	4	A storage allocation error occurred.

Table 44. BLGEXDEL Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
8	8	A database access error occurred.
8	12	The record was not found.
8	16	You cannot delete root VSAM key X'00000000'.
16	4	Internal control blocks were not found.

BLGJAUTH–Check for Authorization

User exit BLGJAUTH determines whether the user's privilege class has the authorization for Tivoli Information Management for z/OS to perform the requested function. This user exit can be used outside the API environment.

Input A 4-character authorization code in the TSCA variable data area.

Output
Return and reason codes as listed in Table 45.

Environment
TSP and TSX

Table 45. BLGJAUTH Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	The user is authorized to perform the requested action.
4	0	The user is not authorized to perform the requested action.
8	0	The variable data length (TSCAVDAL) is not equal to 4.
16	0	Internal control blocks were not found.

BLGTSAPI–Test for API Environment

User exit BLGTSAPI determines if the API environment is active. This user exit can be used outside the API environment.

Input None.

Output
TSCA return code field TSCAFRET.

Environment
TSP and TSX

Possible return codes are listed in Table 46.

Table 46. BLGTSAPI Return Codes

Return Code (TSCAFRET)	Description
0	The API environment is active.

Table 46. BLGTSAPI Return Codes (continued)

Return Code (TSCAFRET)	Description
4	The API environment is not active.

BLGYAPBR–API Record Build Processor

Retrieves data stored in API data structures and converts it to Tivoli Information Management for z/OS internal record form ready to be stored in the database. Create (T102), update (T105), and add record relation (T109) transactions use this exit.

Input API data structures.

Output
Internal form of record. Possible return codes are listed in Table 47.

Environment
TSP and TSX

Table 47. BLGYAPBR Return Codes

Return Code (TSCAFRET)	Description
0	Successful completion.
4	A processing error occurred.

BLGYAPBU–API Retrieve Record ID

Retrieves the record ID or root VSAM key specified in PICA field PICARNID.

- If a record ID is retrieved, it is appended to the TSCA variable data area.
- If a root VSAM key is retrieved, it is converted to a record ID, and the record ID is appended to the TSCA variable data area.

Update (T105), add record relation (T109), and delete (T100) transactions use this exit.

Input PICA field PICARNID

Output
TSCA variable data area. Possible return and reason codes are listed in Table 48.

Environment
TSP and TSX

Table 48. BLGYAPBU Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion. The record ID was placed in the TSCA variable data area.
4	0	An overflow occurred in the variable data area. The data was not added to the TSCA variable data area.

Table 48. BLGYAPBU Return and Reason Codes (continued)

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
4	8	The root VSAM key was placed in the TSCA variable data area. An error occurred in displaying the record.
4	16	The root VSAM key was placed in the TSCA variable data area, but the root VSAM key is not valid.

BLGYAPCP–API Control Processor

With TSP BLGAPI00, user exit BLGYAPCP performs some routing and control processing for the LLAPI. BLGYAPCP processes many transactions within code segments. When a transaction is implemented through a TSP, BLGYAPCP copies the transaction code stored in PICA field PICATRAN left justified to TSCA field TSCAUFLD. BLGYAPCP then returns to TSP BLGAPI00 to complete transaction processing by linking to a transaction TSP. Linking is performed by testing the transaction code passed in TSCAUFLD.

Input None

Output

Transaction code stored in TSCA field TSCAUFLD.

Environment

TSP and TSX

BLGYAPGP–API Retrieve Panel Name and Check for Special Processing

Retrieves the name of a summary panel for use in record create or update processing. BLGYAPGP also checks fields PIDTUSEF, PICATSAU, and PICAHIST to determine whether dynamic PIDT, text audit data, or history data processing was requested. The exit then sets a return and reason code in the TSCA to indicate which functions were requested.

Summary panel names are stored in API record processing control panels BLG1AACP and BLG1AAUP. The value stored in PIDT field PIDTUSEF determines which control panel to use. Each control line of these panels specifies an s-word and a target summary panel name. The record type s-word that was specified in the PIDT used to perform the transaction is used as a scan search argument. This search argument is compared with each control line s-word in the control panel until a match is found. When the control line is found, BLGYAPGP extracts the target panel name, stores it in the TSCA variable data area, and sets the variable data area length TSCAVDAL.

Create (T102), update (T105), and add record relation (T109) transactions use this exit.

Input None

Output

Panel name stored in the TSCA variable data area. Possible return and reason codes are listed in Table 49 on page 268.

Environment
TSP and TSX

Table 49. BLGYAPGP Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion. PIDTUSER≠D, PICATXAU≠Y, PICAHIST≠Y.
4	0	A summary panel s-word was not found, or the application is not authorized to perform the function.
4	1	Successful completion. PIDTUSER≠D, PICATXAU≠Y, PICAHIST=Y.
4	2	Successful completion. PIDTUSER≠D, PICATXAU=Y, PICAHIST≠Y.
4	3	Successful completion. PIDTUSER≠D, PICATXAU=Y, PICAHIST=Y.
4	4	Successful completion. PIDTUSER=D, PICATXAU≠Y, PICAHIST≠Y.
4	5	Successful completion. PIDTUSER=D, PICATXAU≠Y, PICAHIST=Y.
4	6	Successful completion. PIDTUSER=D, PICATXAU=Y, PICAHIST≠Y.
4	7	Successful completion. PIDTUSER=D, PICATXAU=Y, PICAHIST=Y.

BLGYAPSR–API Set Interface Reason Code

Retrieves data from the TSCA variable data area and converts it to a reason code that is passed back in PICA field PICAREAS.

Input TSCA variable data area. If the variable data is 4, the data is converted and put into PICAREAS. If the variable data length is anything but 4, the first 2 characters of the data are converted and put into PICAREAS.

Output
PICA field PICAREAS. The possible return code is listed in Table 50.

Environment
TSP and TSX

Table 50. BLGYAPSR Return Codes

Return Code (TSCAFRET)	Description
0	Completion.

BLGYAPUP–Verify Record Update

User exit BLGYAPUP verifies that the record specified to the interface is being updated. Update (T105) and add record relation (T109) transactions use this exit.

Input None.

Output

TSCA return code field TSCAFRET. Possible return codes are listed in Table 51.

Environment

TSP and TSX

Table 51. BLGYAPUP Return Codes

Return Code (TSCAFRET)	Description
0	The specified record is being updated.
4	The specified record is not being updated.

BLGYITSP–Invoke a TSP or TSX

User exit BLGYITSP can be used in an API environment to invoke a TSP or TSX. The name of the TSP or TSX is contained in the LLAPI control block field PICAUTSP. PICAUTSP is set by the HLAPI with the value of the input PDB name TSP_NAME on the HL14 transaction. A single string of parameter data can also be passed to the TSP or TSX. PICAPARL contains the length of the data pointed to by PICAPARM. If PICAPARL is non-zero and PICAPARM is non-zero, the data pointed to by PICAPARM is passed to a TSP using the variable data area or to a TSX as an argument. The value of PICAPARL must not exceed 255. If PICAUTSP is not set, no TSP or TSX is invoked. With the HLAPI using HL14, you can pass input data to an invoked TSX and get data back from the TSX in the form of output PDBs. The TSX uses GETAPIDATA to access input data and uses SETAPIDATA to return data to the requesting application.

Input None.

Output

TSCA return code field TSCAFRET. Possible return codes are listed in Table 52.

Environment

TSP and TSX

Table 52. BLGYITSP Return Codes

TSCAFRET	TSCAFRES	PICAREAS	Description
0	0	0	Processing successful. The TSP or TSX was invoked successfully.
4	4	n/a	Processing successful. No TSP or TSX specified to invoke (PICAUTSP not specified).
8	4	n/a	Not in API environment. TSP or TSX not invoked.
8	8	164	Error invoking the TSX. The most likely cause is that the TSX cannot be found.

Table 52. BLGYITSP Return Codes (continued)

TSCAFRET	TSCAFRES	PICAREAS	Description
12	4	163	The PICAPARL value is too large.
16	4	164	Error invoking the TSP or TSX. The TSP or TSX starting panel and the current panel are not the same.
16	8	164	Error invoking the TSP or TSX. The TSP or TSX could not be found.
16	12	164	Error invoking the TSP or TSX. The TSP or TSX could not be loaded,
16	16	164	Error invoking the TSP or TSX. Storage error.
16	20	164	Error invoking the TSP. The specified panel is not a TSP.
16	24	164	Error invoking the TSP or TSX.

Configuration Migration User Exits

The following user exits are provided with the configuration migration TSPs. If you are a Version 2 user and have modified your configuration panels, you can use these user exits to modify the TSPs provided with this version.

BLMMIGAE–Add Data Entry

BLMMIGAE adds to the current record a structured data-entry (SDE) that cannot be added using the TSP ADDDATA and PROCESS control lines. Use BLMMIGAE only when you cannot collect the requested entry using normal record processing, such as an entry that is automatically collected only when a record is created.

You can specify the structured word index, the prefix index, or both in the Data Field Specification panel (BLM8CU9P) of the USEREXIT control line.

- Specify the structured word index if the s-word is included in the SDE being added to the record.
- Specify the prefix index if the prefix is included in the SDE being added to the record. You specify the prefixed data as literal data for the SDE added to the record. If no prefix index is specified, the literal data is added to the SDE without a prefix.

Note: Failure to add the entry does not end processing of the record by the migration utility.

Input The information entered in one of the following fields on the Data Field Specification panel:

- **Structured word index**
- **Prefix index**
- **Literal/Test data**.

Output

An SDE added to the current record. Possible return codes are listed in Table 53 on page 271.

Environment
TSP only

Table 53. BLMMIGAE Return Codes

Return Code (TSCAFRET)	Description
0	Successful completion.
4	An error was detected. Message BLM13004 is written.
16	An internal error occurred. Contact your IBM representative.

The BLGMIG21 and BLGMIG39 TSPs contain examples of this user exit.

BLMMIGDD–Delete Dialog

BLMMIGDD deletes from the current record the dialog located by the last FINDSDATA function. The FINDSDATA function must specify the s-word index that begins the dialog to be deleted. Any dialogs embedded in the selected dialog are also deleted.

The dialog is not deleted under the following conditions:
- The TSCA does not contain an s-word (FINDSDATA not processed).
- The record does not contain the s-word.
- The s-word does not begin a dialog.
- There is a dialog structure begin/end mismatch.

You normally use BLMMIGDD after the recursive running of BLMMIGMD, which moves all pertinent responses to another dialog.

Note: Failure to delete the dialog does not end processing of the record by the migration utility.

Input A FINDSDATA processed with a TSCA return code of zero to locate the s-word index of the SDE containing the dialog begin for the dialog to be deleted.

Output
Deletion of the dialog. Possible return codes are listed in Table 54.

Environment
TSP and TSX

Table 54. BLMMIGDD Return Codes

Return Code (TSCAFRET)	Description
0	Successful completion.
4	An error was detected. Message BLM13005 is written.
16	An internal error occurred. Contact your IBM representative.

The BLGMIG3E TSP provides an example of using this user exit.

BLMMIGDE–Delete Data Entry

BLMMIGDE deletes from the current record the SDE located by the last FINDSDATA control line. You usually use BLMMIGDE after BLMMIGMD processing to delete an SDE that is moved to another dialog.

Note: Failure to delete the entry does not end processing of the record by the migration utility.

Input A FINDSDATA control line processed with a return code of zero.

Output
Deletion of the SDE. Possible return codes are listed in Table 55.

Environment
TSP and TSX

Table 55. BLMMIGDE Return Codes

Return Code (TSCAFRET)	Description
0	Successful completion.
4	An error was detected.
16	An internal error occurred. Contact your Tivoli representative.

The BLGMIG3D TSP provides an example of this user exit.

BLMMIGEC–Check Error Flag

BLMMIGEC checks both the migration error status for the current record and the termination status for the migration utility.

When a linked TSP returns control to the calling TSP, the TSCAFRET field is usually checked to determine if migration of the current record should continue. If the TSCAFRET field is zero, migration of the current record continues. If the TSCAFRET field is not zero, no further processing of the current record is attempted, and error recovery procedures are initiated to cancel processing the current record. If the TSCAFRET field is not zero after record error recovery is completed, migration stops.

Input None.

Output
A return code, as listed in Table 56.

Environment
TSP and TSX

Table 56. BLMMIGEC Return Codes

Return Code (TSCAFRET)	Description
0	No error status or termination was set.
4	Record error status/termination error status was set.

The migration utility panel BLGMIG21 provides an example of this user exit.

BLMMIGFC–Passed/Failed Record Count

BLMMIGFC either clears the consecutive record error counter or increments and tests the consecutive record error counter. This counter is used to control detection of a permanent error condition found by the migration utility.

To clear the record error counter and increment the number of records migrated:

Input PASSED in the **Literal/Test data** field of the Data Field Specification panel of the USEREXIT control line calling this user exit.

Output
Error counter is set to zero; records-migrated counter is incremented by 1.

Environment
TSP only

To check the record error counter:

Input FAILED in the **Literal/Test data** field of the Data Field Specification panel of the USEREXIT control line calling this user exit.

Output
Error counter is incremented and checked; migration utility termination status is set if the value of the error counter is greater than the specified threshold.

Environment
TSP only

To reset the record error counter:

Input YES in the **New data** field of the Data Field Specification panel of the USEREXIT control line calling this user exit.

Output
Consecutive error counter is initialized to zero.

Environment
TSP only

Possible return codes are listed in Table 57.

Table 57. BLMMIGFC Return Codes

Return Code (TSCAFRET)	Description
0	Successful completion.
4	An internal error occurred. Contact your Tivoli representative.
8	Internal error threshold was reached. Message BLM13003 was written.
16	An internal error occurred. Contact your Tivoli representative.

The migration utility panels BLGMIG21 and BLGMIG91 show examples of this user exit.

BLMMIGFS–Free Migration Environment

BLMMIGFS closes the environment of the migration utility. It must be the last action in the migration utility. After the BLMMIGFS exit is processed, no other migration utility user exit can be processed.

Input None.

Output
The migration utility environment no longer exists. Possible return codes are listed in Table 58.

Environment
TSP and TSX

Table 58. BLMMIGFS Return Codes

Return Code (TSCAFRET)	Description
0	Successful completion.
4	An internal error occurred. Contact your Tivoli representative.
16	An internal error occurred. Contact your Tivoli representative.

BLMMIGGS–Set Up Migration Environment

BLMMIGGS sets up the environment for the migration utility. All of the migration utility user exits depend upon the environment established by this exit. This exit also determines the consecutive error threshold value.

To set up environment:

Input None.

Output
Environment for migration utility.

Environment
TSP and TSX

To establish the value for consecutive error threshold:

Input The contents of the **New data** field on the Data Field Specification panel of the USEREXIT control line calling this exit.

Output
Error threshold is set; a threshold of 3 is used if the value specified for the **New data** field is not valid or has been omitted.

Possible return codes are listed in Table 59.

Environment
TSP only

Table 59. BLMMIGGS Return Codes

Return Code (TSCAFRET)	Description
0	Successful completion.
8	An out of storage condition occurred.

BLMMIGLC–Error Routine Loop Counter

BLMMIGLC either clears the current record processing counter or increments and tests the current record processing counter. This counter is used during record error

recovery processing. If the current record cannot be canceled within 10 tries, both record error recovery processing and migration utility processing stop.

To clear the loop counter:

Input YES in the **New data** field on the Data Field Specification panel of the USEREXIT control line calling this exit.

Output
Loop counter set to zero.

Environment
TSP only

To increment and test the loop counter:

Input A blank or anything other than YES in the **New data** field of the Data Field Specification panel of the USEREXIT control line calling this exit.

Output
Loop counter is incremented and checked. If the value of the loop counter is greater than 10, migration utility termination status is set.

Environment
TSP only

Possible return codes are listed in Table 60.

Table 60. BLMMIGLC Return Codes

Return Code (TSCAFRET)	Description
0	Successful completion.
4	An internal error occurred. Contact your Tivoli representative.
8	The internal error threshold was reached. Message BLM13001 was written.
16	An internal error occurred. Contact your Tivoli representative.

The migration utility panel BLGMIG91 provides an example of this user exit.

BLMMIGMD–Move Variable Data

BLMMIGMD adds a section of an IRC to the end of the TSCA variable data area. You can use it to move a collected response from one dialog to another or to clear the TSCA variable data area before building an IRC.

To move a collected response to a new dialog, you must first use the FINDSDATA control line to determine if the response is in the current record. Then determine the panel selection number in the new dialog that collects the response. Enter this number in the **Literal/Test data** field on the Data Field Specification panel (BLG8CU9P) of the USEREXIT control line calling this exit. You can precede the selection number in the **Literal/Test data** field with additional data to add to the IRC. The trailing comma after the selection number is added by the user exit and is not specified in the **Literal/Test data** field.

Input The contents of the **Literal/Test data** field on the Data Field Specification panel of the USEREXIT control line calling this exit.

Output
A response is moved to a new dialog. The possible return code is listed in Table 61.

Environment
TSP only

Table 61. BLMMIGMD Return Codes for Moving a Collected Response

Return Code (TSCAFRET)	Description
0	Successful completion.

After a response is moved to the IRC, you can remove it from the current dialog with either the BLMMIGDE or BLMMIGDD user exit. When the old dialog is maintained in the record, use BLMMIGDE to delete the response from the old dialog as each response is moved. When the old dialog is no longer part of the new record, use BLMMIGDD to delete the response.

After the IRC is constructed, you must specify **Get variable data** as YES in the ADDDATA control line, then use a PROCESS control line.

To clear the TSCA variable data area:

Input YES in the **New data** field on the Data Field Specification panel of the USEREXIT control line calling this exit.

Output
No other processing. The possible return code is listed in Table 62.

Environment
TSP only

Table 62. BLMMIGMD Return Codes for Clearing the TSCA Variable Data Area

Return Code (TSCAFRET)	Description
0	Successful completion.

The BLGMIG3D TSP provides an example of this user exit.

BLMMIGSA–Search Argument

BLMMIGSA sets the SEARCH command and additional search criteria in the TSCA variable data area to exclude configuration records already migrated to or created in Versions 3 or 4. The search argument **MIGR/.** is added. Data in the TSCA variable data area is lost.

Input None.

Output
The variable data area containing SEARCH command criteria is set to exclude Versions 3 or 4 migrated/created records. The possible return code is listed in Table 63 on page 277.

Environment
TSP and TSX

Table 63. BLMMIGSA Return Codes

Return Code (TSCAFRET)	Description
0	Successful completion.

The migration utility panel BLGMIG11 provides an example of this user exit.

BLMMIGSE–Set Error Flag

BLMMIGSE sets or clears the record migration error status. The retention of error status is required for later use by the migration TSPs. The TSCA return code field (TSCAFRET) cannot be used because it is normally cleared after each TSP control line is processed. Use the BLMMIGEC user exit to check error status.

Error status is controlled by the value specified in the **Literal/Test data** field on the Data Field Specification panel of the USEREXIT control line. Error status is set when ON is specified. Error status is cleared when OFF is specified. Error status is set whenever a condition is detected that prevents migration of the current record. Setting this status cancels processing for the current record. After processing of the current record is canceled, the error status is cleared.

Migration utility panels BLGMIG4A and BLGMIG91 provide examples of this user exit.

Input The contents of the **Literal/Test data** field in the Data Field Specification panel of the USEREXIT control line calling this exit.

ON To set error status.

OFF To clear error status.

Output
Record migration error status is set or cleared. Possible return codes are listed in Table 64.

Environment
TSP only

Table 64. BLMMIGSE Return Codes

Return Code (TSCAFRET)	Description
0	Successful completion.
4	Migration utility termination status was set.

Database Administration User Exits

The following user exits are used by the Automatic Log Save Facility and the DB2® Extract Facility. Refer to the *Tivoli Information Management for z/OS Program Administration Guide and Reference* for details on these user exits.

BLGUT3EX–Recovery
Loads the Tivoli Information Management for z/OS database with records that were off-loaded by the BLGUT4EX user exit.

BLGUT3WT–Initialize for Receive
Initializes the TSCAVDA and waits for the interval specified in the LOGSAVE record.

BLGUT4EX–Offload a Recovery Data Set
Off-loads the recovery data set (SDLDS) into a sequential data set.

BLGUT4WT–Initialize for Send
Initializes the TSCAVDA and waits for the interval specified in the LOGSAVE record.

BLMSSGEN–SQL Setup, Extract, and Cleanup
Converts Tivoli Information Management for z/OS records to SQL statements for loading into a DB2 database. The user exit has three modes: Setup, Extract, and Cleanup.

Escalation and Notification User Exits

The following user exits are used by the escalation facility and immediate notification. Refer to the *Tivoli Information Management for z/OS Program Administration Guide and Reference* for details on these user exits.

BLGESADD–Increment Counter
Adds 1 to the value in TSCAUFLD.

BLGESCCL–Escalation Cleanup
Deletes the escalation load modules and frees the temporary data set.

BLGESCKE–Check escalation
Checks to see if an escalation job is in process.

BLGESCLR–Clear Control Block
Resets the ESCB fields to binary zeros.

BLGESDAT–Date and Time
Puts the system date and time into the TSCA variable data area.

BLGESDUR–Duration
Locates the date and time fields and determines whether the duration (1, 2, or 3 depending on TSCAUFLD) specified by the current rules record is met for notifying a user ID. A problem record must be in update mode, and the escalation control block must contain the appropriate duration.

BLGESFCB–Free Control Block
Frees the escalation control block. This must be the last user exit processed in the escalation function.

BLGESGCB–Get Control Block Storage
Gets storage for the ESCB and anchors it in the TSCA. This user exit must be called before any other escalation user exit is called.

BLGESGET–Get Control Block Field
Gets a field stored by BLGESPUT, BLGESPUV, or a user-written user exit from the ESCB and stores it in the TSCA variable data area.

BLGESINI–Initialize
Loads the escalation user exits and allocates a temporary data set that you can use with the BLGESNOT user exit to send the escalation message.

BLGESLVL–Level Increment
Increases the escalation level by 1 in the current problem record ESCB, if possible.

BLGESNOT–Notify
Builds the command that notifies a user about a problem record. The command is built and stored in the TSCA variable data area, overlaying what was there.

BLGESPRI–Priority Update
Increments the value associated with an input prefix by the priority adjust amount in the escalation control block. The record must be in update mode.

BLGESPUT–Put TSCA Data in Control Block
Gets data from the TSCA and stores it in the escalation control block. This data can be retrieved by user-exit BLGESGET or a user-written exit routine.

BLGESPUV–Put Variable Data
Gets data from the variable data area or literal data and stores it in the ESCB.

BLGESSCT–Store Criteria
Scans the current rules record for an occurrence of escalation criteria (s-word index 0121) and stores its associated prefix in the next available spot in the ESCB.

BLGESSEA–Get Escalation Criteria
Scans the current rules record for an occurrence of data for each escalation criterion, and adds any criteria found to the variable data area as part of a search argument. The search argument locates problem records that meet the escalation criteria in the rules record.

BLGNSYAL–Allocate Data Set to SYSOUT
Dynamically allocates a data set to SYSOUT with a user-specified destination (TCP/IP SMTP node and ID).

BLGNSYFR–Free SYSOUT Data Set
Frees the SYSOUT data set allocated by user exit BLGNSYAL.

BLGUSERS–Extract Mail Address from USERS Record
Extracts the mail addresses from the USERS record that match an input name or mail alias.

General-purpose User Exits

BLGCURDT–Return Current Date, Current Time, and Current Time Zone
The BLGCURDT user exit returns the current date in external format, the current time in external form, and the current time zone (if UT processing is enabled) in the variable data area. For TSXs, this is the variable TSCAVDA. The values that are returned are separated by blanks.

Input None

Output
The external date, the external time, and the external time zone are returned in the TSCA variable data area (TSP) or in the TSCAVDA variable (TSX).

Environment
TSP and TSX

How to call BLGCURDT from a TSX
You can call user exit BLGCURDT from a TSX using the following syntax:

```
CALL BLGTSX 'USEREXIT','BLGCURDT'
```

How to call BLGCURDT from a TSP

You can call user exit BLGCURDT from a TSP by calling the USEREXIT control line.

BLGEDATE–Convert Internal Date to External Date

The BLGEDATE user exit and the BLGIDATE user exit are provided to enable a TSX or TSP to support a variety of external date formats. Users can specify a preferred external date format from a variety of supported external date formats. You can use these user exits to convert dates to and from the correct format as necessary.

The BLGEDATE user exit converts dates in internal date format (YYYY/MM/DD) to the current user's external date format. The current user is the user currently viewing or working with the record. The current user's external date format is derived from the user's profile. Users can specify an external date preference in the User's date format field in the user profile (in "User and database defaults"). If no preference is specified in the user profile, the external date format used for conversion is the external date format specified on the BLGPARMS DATEFMT keyword for the user's session.

The internal date is passed as a parameter from a TSX or in the variable data area from a TSP.

The user exit does not confirm that a date is valid; it only confirms that the format is valid. For example, an internal date of 2001/05/35 will be successfully converted to external format 05/35/2001, even though the date is not valid.

Input For TSPs, put the internal date to be converted into the TSCA variable data area, then call the USEREXIT control line to invoke this user exit. For TSXs, specify the internal date format as a parameter when calling this user exit.

Output

The external date is returned in the TSCA variable data area (TSP) or in the TSCAVDA variable (TSX).

Environment

TSP and TSX

How to call BLGEDATE from a TSX

You can call user exit BLGEDATE from a TSX using the following syntax:

```
CALL BLGTSX 'USEREXIT','BLGEDATE',intdate
```

Parameter Descriptions:

1. intdate

Valid reply

The date in internal format.

Default

None

Required

How to call BLGEDATE from a TSP

You can call user exit BLGEDATE from a TSP by entering the date to be converted in the variable data area, and then calling the USEREXIT control line.

Table 65. BLGEDATE Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	The date was successfully converted.
8	8	A conversion failure occurred.

BLGIDATE–Convert External Date to Internal Date

The BLGIDATE user exit converts dates in the current user's external date format to 10-character internal date format (YYYY/MM/DD). For a description of the "current user's" date format, see the description of the BLGEDATE user exit.

The external date is passed as a parameter from a TSX or in the variable data area from a TSP.

The user exit does not confirm that a date is valid; it only confirms that the format is valid. For example, an internal date of 2001/05/35 will be successfully converted to external format 05/35/2001, even though the date is not valid.

Input For TSPs, put the internal date to be converted into the TSCA variable data area, then call the USEREXIT control line to invoke this user exit. For TSXs, specify the internal date format as a parameter when calling this user exit.

Output

The internal date is returned in the TSCA variable data area (TSP) or in the TSCAVDA variable (TSX).

Environment

TSP and TSX

How to call BLGIDATE from a TSX

You can call user exit BLGIDATE from a TSX using the following syntax:

```
CALL BLGTSX 'USEREXIT','BLGIDATE',extdate
```

Parameter Descriptions:

1. extdate

Valid reply

The date in external format.

Default

None

Required

How to call BLGIDATE from a TSP

You can call user exit BLGIDATE from a TSP by entering the date to be converted in the variable data area, and then calling the USEREXIT control line.

Table 66. BLGIDATE Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	The date was successfully converted.
8	8	A conversion failure occurred.

BLGJSKIP–Skip Transfer-to or Owning Class Processing

Disables or enables the updating of the owning privilege class when a record is filed. When **Apply not logic** is set to NO, updating of the owning class is disabled. When **Apply not logic** is set to YES, updating of the owning class is enabled.

Input TSCA1ANL, set on or off.

Output

Transfer-to or Owning privilege class processing is disabled or enabled. Possible return and reason codes are listed in Table 67.

Environment

TSP only

Table 67. BLGJSKIP Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	Successful completion.
8	8	A serious processing error occurred.

The BTNTTOTT TSP provides an example of using this user exit.

BLGSPFGT–Retrieve Variable from an ISPF Pool

Copies data from an ISPF pool to the TSCA variable data area. BLGSPFGT first searches the function pool for the variable, then the shared pool, and finally the profile pool.

Input The **Label name** field in the Data Specification Panel of the USEREXIT control line contains the name of the ISPF variable to retrieve.

The **Replace data?** field in the Flag Field Specification Panel of the USEREXIT control line is YES if you want to replace any existing data in the variable data area with the value of the ISPF variable. If this field is NO, any existing data is appended with the value of the ISPF variable.

Output

The variable is retrieved from one of the ISPF profile pools (searched in the order of function pool, shared pool, and finally, profile pool). Possible return and reason codes are listed in Table 68 on page 283.

Environment

TSP and TSX

How to call BLGSPFGT from a TSX

You can call user exit BLGSPFGT from a TSX using the following syntax:

```
CALL BLGTSX 'USEREXIT','BLGSPFGT',varname
```

Parameter Descriptions:

1. varname

Valid reply

The name of the ISPF variable that you wish to get. The value of ″varname″ is stored in the variable data area (VDA).

Note: The contents of the variable data area are replaced with the results from BLGSPFGT.

Default

None

Required

Table 68. BLGSPFGT Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	The ISPF variable was successfully copied to the TSCA variable data area.
0	4	The ISPF variable was found but was empty.
8	4	Insufficient space exists in the variable data area for the variable contents.
8	8	The ISPF VCOPY service did not retrieve the variable because the **Label name** field does not contain a valid ISPF variable name. TSCAFRES is set to the return code from VCOPY (refer to *Dialog Management Services* for the meaning of the return codes).
12		The ISPF VCOPY service did not retrieve the variable. TSCAFRES is set to the return code from VCOPY (refer to *Dialog Management Services* for the meaning of the return codes).

The BLGTPSET TSP provides an example of using this user exit.

BLGSPFPT–Update Variable in the ISPF Profile Pool

Copies data from the literal or variable data field to the ISPF profile pool. If a variable of the same name exists in the profile pool, it is updated. If it does not exist in the pool, it is created.

Note: The user exit **BLGSPFPT** only does a VPUT to the profile pool. Any other instances of the variable that exist in either the function pool or the shared pool are left unchanged.

Input The **Label name** field in the Data Specification Panel of the USEREXIT control line containing the name of the ISPF variable to be created or updated.

- If you want to get the data from the variable data area, the **Get variable data** field in the Flag Field Specification Panel of the USEREXIT control line set to YES.
- If you want to use literal data, the **Literal/test data** field in the Data Specification Panel of the USEREXIT control line containing a value.

- If you want to clear out the variable, the **Get variable data** field set to NO. The **Literal/test data** field should be blank.

Output

The ISPF variable is set to the specified value. Possible return and reason codes are listed in Table 69.

Environment

TSP and TSX

How to call BLGSPFPT from a TSX

You can call user exit BLGSPFPT from a TSX using the following syntax:

```
CALL BLGTSX 'USEREXIT','BLGSPFPT',varname,value
```

Parameter Descriptions:

1. varname

Valid reply

The name of the ISPF variable that you wish to set.

Default

None

Required

2. value

Valid reply

The value that you want to set the variable to.

Default

None

Required

Table 69. BLGSPFPT Return and Reason Codes

Return Code (TSCAFRET)	Reason Code (TSCAFRES)	Description
0	0	The profile pool was successfully updated.
8	4	**Literal/test data** cannot be specified when **Get variable data** is set to YES.
8	8	The **Label** field does not contain a variable.
12		The ISPF VPUT service did not end successfully. TSCAFRES was set to the return code from VPUT (refer to *Dialog Management Services* for the meaning of the return codes).

The BLGTPSET TSP provides an example of using this user exit.

BLMXSPRM–Provide Values of Session Parameters

This user exit copies data from the TPCB, DCDT, and DSAT control blocks into the TSCA variable data area. It returns the values of these session parameters:

- Default external date format (DATEFMT keyword)
- Default time zone (TIMEZONE keyword)
- Date conversion routine name (DATECNV keyword)
- Time conversion routine name (TIMECNV keyword)
- Data model database ID and trigger character (MODELDB keyword)
- Start panel name (PANEL keyword)
- "Old record" external date form (ODATEFMT keyword)
- "Old record" time zone (OTIMEZON keyword)
- Session member suffix (the last two characters of the session member name)
- Date and time the session member was last assembled
- Subsystem name (CAS keyword)
- Sort routine name (SORT keyword)
- Type of sort used for search results sorting (EXTSORT keyword)
- Attention key status (ATTNKEY keyword)
- Search/sort record limits (SORTPFX keyword)
- Trigger character for multi-cluster SDDS/SDIDS (BLGCLUST TRIGGER keyword)
- Data set name of the dictionary data set
- Data set name or DDNAME of the RFT data set, or both
- Number of SDDS clusters and data set name of the first cluster
- Number of SDIDS clusters and data set name of the first cluster
- Data set name of the SDLDS, if any
- Label, data set name, and access type (read-only or write) of up to seven red panel data sets

These values are returned in a fixed format in the variable data area (TSCAVDA variable).

The Desktop calls the BLMXSPRM user exit from TSX BLGTDFDT to access default date and time formats. To see an example of this, view TSX BLGTDFDT in the TSX dataset.

Note: The BLMXSPRM user exit provides a function similar to that of the BLGTSESS TSX. The difference is that the BLMXSPRM user exit puts values in the variable data area and the BLGTSESS TSX uses REXX SAY statements to show the session member values on the user's screen.

Input To use the BLMXSPRM user exit, issue a TSX USEREXIT control line. Optionally, you can include a parameter to indicate what type of data to retrieve. Then parse the value returned in the TSCAVDA variable to access the desired values.

Output
The output from user exit BLMXSPRM is a single character string that

varies in format depending on whether you choose the GENERAL option or the PANELS option. The following structure occurs when you specify the GENERAL option:

```
  Name   Length   Macro    Keyword        Description
-------- ------  --------  --------  ------------------------
DFLTDFMT   10    BLGPARMS  DATEFMT   Default date format
DFLTTZON    8    BLGPARMS  TIMEZONE  Default time zone
DATERTN     8    BLGPARMS  DATECNV   Date conversion routine
TIMERTN     8    BLGPARMS  TIMECNV   Time conversion routine
DMDLDB      1    BLGPARMS  MODELDB   Data model database ID
DMDLTRIG    1    BLGPARMS  MODELDB   Data model trigger char
STARTPNL    8    BLGPARMS  PANEL     Start panel
ORECDFMT   10    BLGPARMS  ODATEFMT  "Old record" date format
ORECTZON    8    BLGPARMS  OTIMEZON  "Old record" time zone
SESSID      2                        Session member suffix
ASMDATE     8                        Assembly date
ASMTIME     5                        Assembly time (HH:MM)
SUBSYS      4    BLGPARMS  CAS       Subsystem ID of BLX-SP
SORTRTN     8    BLGPARMS  SORT      Sort routine name
SRCHSORT    1    BLGPARMS  EXTSORT   Sort routine for search
                                     E=External, I=Internal
ATTNKEY     1    BLGPARMS  ATTNKEY   Attention key setting
                                     D=Disabled, E=Enabled
SORTPFX1   10    BLGPARMS  SORTPFX   Sort prefix #1 value
SORTPFX2   10    BLGPARMS  SORTPFX   Sort prefix #2 value
SORTPFX3   10    BLGPARMS  SORTPFX   Sort prefix #3 value
DICTDSN    44    BLGCLDSN  DSN       Dictionary data set name
RFTDSN     44    BLGCLDSN  DSN       RFT data set name
RFTDDN      8    BLGCLDSN  FILE      RFT DD name
DBTRIG      1    BLGCLUST  TRIGGER   Database trigger char
SDDSCNT     3    BLGCLUST  TRIGGER   Number of SDDSs
SDDSDSN    44    BLGCLDSN  DSN       SDDS #1 data set name
SDIDSCNT    3    BLGCLUST  TRIGGER   Number of SDIDSs
SDIDSDSN   44    BLGCLDSN  DSN       SDIDS #1 data set name
SDLDSDSN   44    BLGCLDSN  DSN       SDLDS data set name
```

The following structure occurs when you specify the PANELS option:

```
 Name   Length   Macro    Keyword        Description
-------- ------  --------  --------  ------------------------
WRITELBL    8    BLGCLDSN  (label)   Write panel data set label
WRITEDSN   44    BLGCLDSN  DSN       Write panel data set name
*          10                        Reserved for expansion
READCNT     2                        # of read panel data sets
(The following is repeated for up to 7 read panel data sets.)
READLBL     8    BLGCLDSN  (label)   Read panel data set label
READACC     1    BLGCLDSN  RDONLY    Read panel data set access (R or W)
READDSN    44    BLGCLDSN  DSN       Read panel data set name
*          11                        Reserved for expansion
```

Note: The values in the Name columns above are the session parm keywords these values are associated with.

Environment

TSX

How to call BLMXSPRM from a TSX

You can call user exit BLMXSPRM from a TSX using the following syntax:

```
CALL BLGTSX 'USEREXIT','BLMXSPRM',option
```

Parameter Descriptions:

1. option

Valid reply

Specify GENERAL for all information except panel data sets.
Specify PANELS for panel data set information.

Default

None

Required

A

Terminal Simulator Communications Fields

The Terminal Simulator Communications Area (TSCA) is a control block that contains flag indicator fields and data fields. Tivoli Information Management for z/OS initializes this control block when a TSP is called.

The TSCA is shipped with the product and is an assembler DSECT named BLGUTSCA. You can use the assembler copy or build your own.

You can also modify TSCA fields. However, some fields, such as the variable data area pointer field, cause problems if they are modified. The tables in this section show which fields not to modify.

Table 70 and Table 71 on page 294 describe the contents of the control block, including the field lengths and the offset of each field into the record. The last three columns of the tables indicate the following:

ASU The field is automatically updated when control returns to the TSP.
NUS Do not change the field using an exit routine.
US You can change the field using an exit routine.

Contents of the TSCA

Part of the SBLMMACS data set that is shipped with Tivoli Information Management for z/OS is an assembler DSECT of the TSCA. Its member name is BLGUTSCA. Table 70 describes the fields in BLGUTSCA that are set by Tivoli Information Management for z/OS or a user exit when a control line is processed.

Table 70. Contents of the TSCA

Field Name	Description	Field Length (bytes)	Hex Offset	ASU	NUS	US
TSCAACRN - Acronym	TSCA	4	0		X	
TSCAFRET - Function return code	Set by TSP control lines to show processing results.	4	4	X		X
TSCAFRES - Function reason code	Set by TSP control lines to show processing results.	4	8	X		X

Table 70. Contents of the TSCA (continued)

Field Name	Description	Field Length (bytes)	Hex Offset	ASU	NUS	US
TSCACPNL - Current panel name	Name of the Tivoli Information Management for z/OS panel currently being processed. For help panels, this field is set to BLG1T007.	8	C	X	X	
TSCALFID - Last filed record ID	Record ID of the most recently filed record. This field contains blanks if no record has been filed.	8	14	X	X	
TSCACRID - Current record ID	Record ID of the record currently being accessed. This field contains blanks if no record has been accessed.	8	1C	X	X	
TSCAUFLD - User field	Set when a SETFIELD or USEREXIT control line is processed.	8	24			X
TSCAMSGC - Total messages	Total number of messages on the current or system message chain. This number includes any messages generated by the previous PROCESS control line or by a MESSAGE control line run since the last PROCESS control line when the **Save generated message** field was not set to YES.	4	2C	X	X	
TSCACPOS - Find string location	Byte number of the position at which a string is found when a find-string-anywhere operation is performed for a TESTFIELD.	4	30	X		
TSCACTPL	Current table panel line.	272	34			
TSCATPLC - Total line count	Number of lines on the current table panel.	4	34	X	X	
TSCATPLN - Current line number	Number (relative to 1) of the current line on the current table panel.	4	38	X	X	
TSCAMTBL - Maximum table panel line length	Longest line length that is allowed for the current line of the current table panel.	4	3C	X	X	
TSCACTBL - Current table panel line length	Amount of data that is in the current table panel line.	4	40	X		

Table 70. Contents of the TSCA (continued)

Field Name	Description	Field Length (bytes)	Hex Offset	ASU	NUS	US
TSCATBLL - Current table panel line	Data to be entered on the current line of the current table panel. When a PROCESS control line runs, the current line of the table panel is processed. The data in this field becomes the current line, and Tivoli Information Management for z/OS copies the next line of data into this field.	256	44	X		X
TSCAVDA	Variable data area.	12	144			
TSCAVDAM - Maximum user variable data length	Set at TSP initialization. The length of the buffer obtained is equal to the length of the command-line reply buffer. This field should be checked by an exit routine that is modifying the user variable-data area to verify that TSCAVDAL does not exceed its length.	4	144		X	
TSCAVDAL - Current user variable data length	Contains the length of the data currently in the user variable-data area. Several functions use this field to determine how much data from the user variable-data area is to be used. Set by a USEREXIT control line.	4	148			X
TSCAVDAP - User variable data area pointer	Set at TSP initialization and released when the TSP ends. Exit routines are responsible for moving data into this data area for get-variable-data operations.	4	14C		X	
TSCAFB	Flatten buffer information.	8	150			
TSCAFBL - Flatten buffer length	Length of the flatten buffer as specified by TSCAFBP.	4	150		X	
TSCAFBP - Pointer to flattened record	Address of the area containing the flattened record produced by the FLATTEN control line. The first 2 bytes of the record contain the length of the flattened record, including the length field itself.	4	154		X	
TSCAUFB	Unflatten buffer information.	8	158			

Table 70. Contents of the TSCA (continued)

Field Name	Description	Field Length (bytes)	Hex Offset	ASU	NUS	US
TSCAUFBL - Unflatten buffer length	Contains the length of the unflatten buffer specified by TSCAUFBP.	4	158			X
TSCAUFBP - Pointer to unflattened record	Contains the address of the area containing the unflattened record retrieved by a user program for the UNFLATTEN control line. The first 2 bytes of the record contain the length of the flattened record (the record to be unflattened), including the length field itself.	4	15C			X
TSCAUPTR - User anchor field	Full-word pointer field reserved for the user. You can use this field to point to a user GETMAINed control block. A user exit can also set this field to a value that it or another user exit can interpret.	4	160			X
TSCAVPHR	FINDSDATA and FINDSJRNL information.	20	164			
TSCAVPHL - Visible phrase length	Length of the visible phrase specified by TSCAVPH. This field is set by the FINDSDATA control line.	4	164	X		
TSCAVPH - Visible phrase	Visible phrase associated with the item found by the FINDSDATA control line, such as RECS=PROBLEM. This field has a value only when the found data was collected from a selection panel, an options panel, or a data-entry panel.	16	168	X		
TSCARSWD	S-Word information for FINDSDATA and FINDSJRNL.	14	178			
TSCARSDL - S-Word length	Length of the s-word specified by TSCARSD. This field is set by either the FINDSDATA or FINDSJRNL control line.	4	178	X		
TSCARSD - S-Word	S-Word associated with the item found by the FINDSDATA, FINDSJRNL, or READDICT control line.	10	17C	X		
TSCARRBL	Data length of the residual reply buffer.	2	186	X	X	

Table 70. Contents of the TSCA (continued)

Field Name	Description	Field Length (bytes)	Hex Offset	ASU	NUS	US
TSCARPWD	Prefix information for FINDSDATA and FINDSJRNL.	14	188			
TSCARPDL - Prefix length	Length of the prefix associated with the item found by the FINDSDATA or FINDSJRNL control line, up to and including the / or _ delimiters.	4	188	X		
TSCARPD - Prefix	Prefix associated with the item found by the FINDSDATA, FINDSJRNL, or READDICT control line. For example, AB/ (the prefix for ABEND code).	10	18C	X		
TSCASESS	Current session member suffix.	2	196		X	
TSCASDFL - Data field length	Length of the data found during the processing of a FINDSDATA or FINDSJRNL control line.	4	198	X		
TSCASDF - Structured data field	Data associated with a prefix for FINDSDATA, FINDSJRNL, or READDICT.	220	19C	X		
TSCASUSP	Suspension level of the current session.	2	278	X	X	
TSCAPRIV	Name of the current privilege class.	8	27A	X	X	
TSCATLIX	Last used or found index value.	2	282	X		X
TSCARSV9	Reserved	8	284		X	
TSCAMALP	Tivoli control block	4	28C	X	X	
TSCASUBP	Pointer to SUB.	4	290		X	
TSCAIPTR	Tivoli control block	4	294	X	X	
TSCARSVA	Reserved	3	298		X	
TSCALFDB	Database ID of the last filed record when TSCALFID (last filed record ID) is updated.	1	29B	X	X	
TSCACMOF - Accumulated data length	Current length of the data in the command-line reply buffer.	4	29C	X	X	
TSCACMRB - Command-line reply buffer	Where data is accumulated to be passed to Tivoli Information Management for z/OS for processing.	512	2A0		X	

Table 70. Contents of the TSCA (continued)

Field Name	Description	Field Length (bytes)	Hex Offset	ASU	NUS	US
TSCARRB - Residual reply buffer	Area containing the current contents of the Tivoli Information Management for z/OS reply buffer. Under normal conditions, this area contains blanks. When Tivoli Information Management for z/OS finds an error (after a PROCESS control line), the response that is in error, and all unprocessed data from the Tivoli Information Management for z/OS reply buffer, are in this field.	512	4A0	X	X	
TSCACLIN	This area contains fields that contain the current control-line information. If no data exists for a field, it is either blank or set to zero. Table 71 shows each of the fields in TSCACLIN.	278	6A0		X	

Table 71 describes the portion of the TSCA control block (TSCACLIN) that contains the information for each data item collected by the TSP control lines. When a control line is called, and before it is processed, the data from the TSP control line is copied into these fields. A field is either blank or zero if no data exists for it. These fields are for internal TSP processing; none of them should be set by the exit routines.

Table 71. TSCA Control Block and TSP Control Lines

Field Name	Description	Field Length (bytes)	Hex Offset	ASU	NUS	US
TSCAFUNC	Function name	10	6A0	X	X	
TSCA0FLG		1	6AA	X	X	
TSCA0VAR	Get variable data	1 bit	6AA.0	X	X	
TSCA0FST	Find data - first occurrence	1 bit	6AA.1	X	X	
TSCA0LST	Find data - last occurrence	1 bit	6AA.2	X	X	
TSCA0FLO	Test panel or message	1 bit	6AA.3	X	X	
TSCA0FND	Find string anywhere	1 bit	6AA.4	X	X	
TSCA0TRO	Trace control lines	1 bit	6AA.5	X	X	
TSCA0TRL	Trace link function	1 bit	6AA.6	X	X	
TSCAMODL	Tivoli or user exit routine	1 bit	6AA.7	X	X	

Table 71. TSCA Control Block and TSP Control Lines (continued)

Field Name	Description	Field Length (bytes)	Hex Offset	ASU	NUS	US
TSCA1FLG		1	6AB	X	X	
TSCA1ANL	Apply not logic	1 bit	6AB.0	X	X	
TSCA1STG	Treat as string data	1 bit	6AB.1	X	X	
TSCA1PNL	Print current panel	1 bit	6AB.2	X	X	
TSCA1MSG	Print message chain	1 bit	6AB.3	X	X	
TSCA1TSC	Print TSCA	1 bit	6AB.4	X	X	
TSCA1CRD	Flatten current record	1 bit	6AB.5	X	X	
TSCA1LRD	Flatten last record filed	1 bit	6AB.6	X	X	
TSCA1RET	Unflatten, retain record ID	1 bit	6AB.7	X	X	
TSCA2FLG		1	6AC	X	X	
TSCA2UMS	User Notify type message	1 bit	6AC.0	X	X	
TSCA2SGM	Save generated message	1 bit	6AC.1	X	X	
TSCA2IDT	Insert data type	1 bit	6AC.2	X	X	
TSCA2REP	Replace data	1 bit	6AC.3	X	X	
TSCA2GLX	Indicates list item field	1 bit	6AC.4	X	X	
TSCA2PRV	Find previous occurrence	1 bit	6AC.5	X	X	
TSCA2CSS	Mixed case search	1 bit	6AC.6	X	X	
TSCA2RGM	Return generated message	1 bit	6AC.7	X	X	
TSCA3FLG		1	6AD	X	X	
TSCASHDR	S-word information	13	6AE	X	X	
TSCASIX	S-word index	2	6AE	X	X	
TSCASWDL	S-word data length	1	6B0	X	X	
TSCASWD	S-word data	10	6B1	X	X	
TSCARSV1		1	6BB	X	X	
TSCAPHDR		9	6BC	X	X	
TSCAPIX	P-word index	2	6BC	X	X	
TSCAPFXL	Prefix data length	1	6BE	X	X	
TSCAPFX	Prefix data	6	6BF	X	X	
TSCARSV2	Reserved	1	6C5	X	X	
TSCAVHDR		33	6C6	X	X	
TSCAVALL	Validation data length	1	6C6	X	X	
TSCAVAL	Validation data	32	6C7	X	X	
TSCARSV3	Reserved	1	6E7	X	X	
TSCANHDR		13	6E8	X	X	
TSCANSIX	New s-word index	2	6E8	X	X	
TSCANSDL	New s-word data length	1	6EA	X	X	
TSCANSWD	New s-word data	10	6EB	X	X	
TSCARSV4		1	6F5	X	X	
TSCATFLD		14	6F6	X	X	
TSCAFLD	TSCA field name	8	6F6	X	X	
TSCAATTR	TSCA field attribute	1	6FE	X	X	
TSCAMASK	TSCA field name	1	6FF	X	X	
TSCAFLEN	TSCA field length	2	700	X	X	
TSCAOFFS	TSCA field offset	2	702	X	X	

Table 71. TSCA Control Block and TSP Control Lines (continued)

Field Name	Description	Field Length (bytes)	Hex Offset	ASU	NUS	US
TSCA5FLG		1	704	X	X	
TSCAVDAO	Use variable data for output	1 bit	704.0	X	X	
TSCASVAL	Skip assisted-entry validation	1 bit	704.1	X	X	
TSCAUCTL	Use specified control data	1 bit	704.2	X	X	
TSCAPCGZ	Only cognize p-word data	1 bit	704.3	X	X	
TSCACGNZ	Cognize data	1 bit	704.4	X	X	
TSCAAHEA	Journal data	1 bit	704.5	X	X	
TSCAAHEF	Journal sequence	1 bit	704.6	X	X	
TSCAXLEN	TESTFIELD length match	1 bit	704.7	X	X	
TSCAIFLL	User data length	1	705	X	X	
TSCALHDR		33	706	X	X	
TSCALITL	Literal/test data length	1	706	X	X	
TSCALIT	Literal/test data	32	707	X	X	
TSCANLHD		33	727	X	X	
TSCANDAL	New data length	1	727	X	X	
TSCANDAT	New data	32	728	X	X	
TSCALABE		10	748	X	X	
TSCATNUM	Target control line of branch	2	748	X	X	
TSCALABL	Label name	8	74A	X	X	
TSCAPANL	Panel name	8	752	X	X	
TSCAFUEX	Function exit	8	75A	X	X	
TSCAVNAM	Verify name	8	762	X	X	
TSCAIFLD	User data	8	76A	X	X	
TSCANPHD		9	772	X	X	
TSCANPIX	New prefix index	2	772	X	X	
TSCANPFL	New prefix length	1	774	X	X	
TSCANPFX	New prefix data	6	775	X	X	
TSCA6FLG		1	77B			
TSCABCON	Convert to binary	1 bit	77B.0	X	X	
TSCADCON	Convert to decimal	1 bit	77B.1	X	X	
TSCAHCON	Convert to hex	1 bit	77B.2	X	X	
TSCAPNEW	Prefix in new data	1 bit	77B.3	X	X	
TSCADELS	Delete s-word	1 bit	77B.4	X	X	
TSCADELD	Delete data	1 bit	77B.5	X	X	
TSCAFFTX	Text option	1 bit	77B.6	X	X	
TSCARSV7	Reserved	1 bit	77B.7	X	X	
TSCANVHD		33	77C	X	X	
TSCANVLL	New validation data length	1	77C	X	X	
TSCANVAL	New validation data	32	77D	X	X	
TSCARSV6	Reserved	1	79D	X	X	
TSCALIX	List index value	2	79E	X	X	
TSCAPTCH	Reserved	22	7A0	X	X	

TSCA Index

The following is an alphabetical list of TSCA fields with their corresponding hex offsets.

Mapping of the TSCA

The following is an assembler language mapping of the terminal simulator communications area (TSCA). This macro is shipped under the name BLGUTSCA. You can use this mapping, or create your own.

```
*****************************************************************
*
* TERMINAL SIMULATOR COMMUNICATIONS AREA
*
*****************************************************************
TSCA     DSECT                  * TERMINAL SIMULATOR COMM AREA

         DS    0F
TSCAACRN DS    CL4              * TSCA ACRONYM
TSCAFRET DS    F                * FUNCTION RETURN CODE
TSCAFRES DS    F                * FUNCTION REASON CODE
TSCACPNL DS    CL8              * THE CURRENT PANEL NAME
TSCALFID DS    CL8              * THE RECORD ID OF LAST FILED RECORD
TSCACRID DS    CL8              * THE RECORD ID OF CURRENT RECORD
TSCAUFLD DS    CL8              * USER-DEFINABLE FIELD
TSCAMSGC DS    F                * THE # OF MESSAGES ON MESSAGE CHAIN
TSCACPOS DS    F                * BYTE # OF STRING FOUND IN TESTFIELD
TSCACTPL DS    0CL272           * CURRENT TABLE-PANEL LINE INFORMATION
TSCATPLC DS    F                * # OF LINES ON THE CURRENT TABLE PANEL
TSCATPLN DS    F                * LINE # OF CURRENT LINE ON TABLE PANEL
TSCAMTBL DS    F                * MAX. DATA LENGTH ALLOWED FOR TSCATBLL
TSCACTBL DS    F                * CURRENT LENGTH OF DATA IN TSCATBLL
TSCATBLL DS    CL256            * CURRENT TABLE PANEL LINE
TSCAVDA  DS    0CL12            * VARIABLE DATA AREA
TSCAVDAM DS    F                * MAX. LENGTH OF VARIABLE DATA AREA
TSCAVDAL DS    F                * CURRENT LENGTH OF VARIABLE DATA AREA
TSCAVDAP DS    A                * ADDRESS OF THE VARIABLE DATA AREA
TSCAFB   DS    0CL8             * FLATTEN BUFFER INFORMATION
TSCAFBL  DS    F                * SIZE OF THE FLATTEN BUFFER
TSCAFBP  DS    A                * POINTER TO FLATTEN BUFFER
TSCAUFB  DS    0CL8             * UNFLATTEN BUFFER INFORMATION
TSCAUFBL DS    F                * SIZE OF THE UNFLATTEN BUFFER
TSCAUFBP DS    A                * POINTER TO UNFLATTEN BUFFER
TSCAUPTR DS    A                * FULL-WORD POINTER RESERVED FOR USER

*****************************************************************
*
* FIELDS TSCAVPHR, TSCARSWD, TSCARPWD, TSCASDFL, AND TSCASDF
* ARE USED BY THE FINDSJRNL AND FINDSDATA FUNCTIONS
*
*****************************************************************
TSCAVPHR DS    0CL20            * VISIBLE PHRASE RETURNED BY FSD, FSJ
TSCAVPHL DS    F                * LENGTH OF THE VISIBLE PHRASE
TSCAVPH  DS    CL16             * THE ASSOCIATED VISIBLE PHRASE
TSCARSWD DS    0CL14            * S-WORD RETURNED BY FINDSDATA, FINDSJRNL
TSCARSDL DS    F                * ACTUAL RETURNED S-WORD LENGTH
TSCARSD  DS    CL10             * ACTUAL RETURNED S-WORD VALUE
TSCARRBL DS    H                * LENGTH OF RESIDUAL-REPLY BUFFER
TSCARPWD DS    0CL14            * P-WORD RET. BY FINDSDATA, FINDSJRNL
TSCARPDL DS    F                * ACTUAL RETURNED P-WORD LENGTH
TSCARPD  DS    CL10             * ACTUAL RETURNED P-WORD VALUE
TSCASESS DS    CL2              * CURRENT SESSION MEMBER SUFFIX
TSCASDFL DS    F                * LENGTH OF STRUCTURED DATA
TSCASDF  DS    CL220            * STRUCTURED DATA TEXT OR S/PWORD DATA
TSCASUSP DS    CL2              * SUSPENSION LEVEL
TSCAPRIV DS    CL8              * CURRENT PRIVILEGE CLASS
TSCATLIX DS    CL2              * LIST INDEX VALUE
TSCARSV9 DS    CL8              * RESERVED
TSCAMALP DS    A                * TIVOLI CONTROL BLOCK POINTER
TSCASUBP DS    A                * POINTER TO SEARCH USER BLOCK
TSCAIPTR DS    A                * TIVOLI CONTROL BLOCK POINTER
TSCARSVA DS    CL3              * RESERVED
TSCALFDB DS    CL1              * DATABASE OF LAST FILED RECORD
TSCACMOF DS    F                * OFFSET IN COMMAND BUFFER NEXT BYTE
TSCACMRB DS    2CL256           * COMMAND LINE REPLY BUFFER
TSCARRB  DS    2CL256           * RESIDUAL REPLY BUFFER
```

```
*****************************************************************************
*
* TERMINAL SIMULATION CONTROL
*
*****************************************************************************
TSCACLIN DS    0CL278            * CONTROL LINES
TSCAFUNC DS    CL10              * FUNCTION NAME
TSCA0FLG DS    XL1               * CONTROL FLAG "0"
TSCA0VAR EQU   X'80'             * A: GET VARIABLE DATA
TSCA0FST EQU   X'40'             * B: FIND FIRST OR NEWEST OCC FSD, FJR
TSCA0LST EQU   X'20'             * C: FIND LAST OR OLDEST OCC FSD, FJR
TSCA0FLO EQU   X'10'             * D: TESTFLOW: 0=PANEL 1=MESSAGE
TSCA0FND EQU   X'08'             * E: FIND STRING ANYWHERE
TSCA0TRO EQU   X'04'             * F: TRACE CONTROL LINES
TSCA0TRL EQU   X'02'             * G: TRACE LINK FUNCTION
TSCAMODL EQU   X'01'             * H: WHEN SET USER ELSE TIVOLI EXIT
TSCA1FLG DS    XL1               * CONTROL FLAG "1"
TSCA1ANL EQU   X'80'             * I: APPLY NOT LOGIC
TSCA1STG EQU   X'40'             * J: TREAT AS STRING DATA
TSCA1PNL EQU   X'20'             * K: PRINT THE CURRENT PANEL
TSCA1MSG EQU   X'10'             * L: PRINT THE MESSAGE CHAIN
TSCA1TSC EQU   X'08'             * M: PRINT THE TSCA
TSCA1CRD EQU   X'04'             * N: FLATTEN CURRENT RECORD
TSCA1LRD EQU   X'02'             * O: FLATTEN LAST RECORD FILED
TSCA1RET EQU   X'01'             * P: UNFLATTEN RETAIN RECORD ID
TSCA2FLG DS    XL1               * CONTROL FLAG "2" - RESERVED
TSCA2RS1 EQU   X'80'             * RESERVED
TSCA2SGM EQU   X'40'             * R: SAVE GENERATED MESSAGE
TSCA2IDT EQU   X'20'             * S: INSERT DATA TYPE, HEX=ON
TSCA2REP EQU   X'10'             * REPLACE DATA
TSCA2GLX EQU   X'08'             * GET LIST INDEX
TSCA2PRV EQU   X'04'             * FIND PREVIOUS (FINDSDATA)
TSCA2CSS EQU   X'02'             * MIXED CASE SEARCH?
TSCA2RSV EQU   X'01'             * RESERVED
TSCA3FLG DS    XL1               * CONTROL FLAG "3" - RESERVED
TSCASHDR DS    0CL13             * S-WORD INFORMATION
TSCASIX  DS    H                 * S-WORD INDEX
TSCASWDL DS    XL1               * S-WORD DATA LENGTH
TSCASWD  DS    CL10              * S-WORD DATA
TSCARSV1 DS    XL1               * RESERVED
TSCAPHDR DS    0CL9              * P-WORD INFORMATION
TSCAPIX  DS    H                 * P-WORD INDEX
TSCAPFXL DS    XL1               * PREFIX LENGTH
TSCAPFX  DS    CL6               * ACTUAL PREFIX
TSCARSV2 DS    XL1               * RESERVED
TSCAVHDR DS    0CL33             * VALIDATION DATA INFORMATION
TSCAVALL DS    XL1               * VALIDATION DATA LENGTH
TSCAVAL  DS    CL32              * VALIDATION DATA
TSCARSV3 DS    XL1               * RESERVED
TSCANHDR DS    0CL13             * NEW S-WORD INFORMATION
TSCANSIX DS    H                 * NEW S-WORD INDEX
TSCANSDL DS    XL1               * NEW S-WORD DATA LENGTH
TSCANSWD DS    CL10              * NEW S-WORD DATA
TSCARSV4 DS    XL1               * RESERVED
TSCATFLD DS    0CL14             * TSCA FIELD INFORMATION
TSCAFLD  DS    CL8               * TSCA FIELD NAME
TSCAATTR DS    XL1               * FIELD ATTRIBUTE - SEE CONSTANT
TSCAMASK DS    XL1               * MASK OFFSET TO TSCA BIT
TSCAFLEN DS    H                 * TSCA FIELD LENGTH
TSCAOFFS DS    H                 * OFFSET TO TSCA FIELD
TSCA5FLG DS    XL1               * CONTROL FLAG "5"
TSCAVDAO EQU   X'80'             * USE VARIABLE DATA FOR OUTPUT
TSCASVAL EQU   X'40'             * SKIP VALIDATION FOR ASSISTED-ENTRY PANEL
TSCAUCTL EQU   X'20'             * USE CONTROL DATA
TSCAPCGZ EQU   X'10'             * COGNIZE ONLY P-WORD
TSCACGNZ EQU   X'08'             * COGNIZE THE DATA
TSCAAHEA EQU   X'04'             * JOURNAL THIS ITEM
```

```
TSCAAHEF EQU   X'02'             * JOURNAL THIS ITEM FIRST
TSCAXLEN EQU   X'01'             * TESTFIELD MATCH EXACT LENGTH
TSCAIFLL DS    XL1               * SETFIELD FIELD LENGTH
TSCALHDR DS    0CL33             * LITERAL/TEST DATA INFORMATION
TSCALITL DS    XL1               * LITERAL/TEST DATA LENGTH
TSCALIT  DS    CL32              * LITERAL/TEST DATA
TSCANLHD DS    0CL33             * NEW DATA INFORMATION
TSCANDAL DS    XL1               * NEW DATA LENGTH
TSCANDAT DS    CL32              * NEW DATA
TSCALABE DS    0CL10             * CONTROL LINE LABEL INFORMATION
TSCATNUM DS    H                 * TARGET CONTROL LINE OF BRANCH
TSCALABL DS    CL8               * LABEL NAME
TSCAPANL DS    CL8               * PANEL NAME/MESSAGE ID
TSCAFUEX DS    CL8               * FUNCTION EXIT NAME
TSCAVNAM DS    CL8               * VERIFY NAME
TSCAIFLD DS    CL8               * PANEL INPUT FROM SETFIELD
TSCANPHD DS    0CL9              * NEW P-WORD INFORMATION
TSCANPIX DS    XL2               * NEW P-WORD INDEX
TSCANPFL DS    XL1               * NEW P-WORD LENGTH
TSCANPFX DS    CL6               * ACTUAL NEW PREFIX
TSCACONV DS    XL1               * DATA CONVERSION
TSCABCON EQU   X'80'             * CONVERT TO BINARY (MOVEVAR)
TSCADCON EQU   X'40'             * CONVERT TO DECIMAL (MOVEVAR)
TSCAHCON EQU   X'20'             * CONVERT TO HEX (MOVEVAR)
TSCAPNEW EQU   X'10'             * PREFIX IN NEW DATA
TSCARSV7 EQU   X'0F'             * RESERVED
TSCANVHD DS    0CL33             * NEW VALIDATION DATA INFORMATION
TSCANVLL DS    XL1               * NEW VALIDATION DATA LENGTH
TSCANVAL DS    CL32              * NEW VALIDATION DATA
TSCARSV6 DS    XL1               * RESERVED
TSCALIX  DS    CL2               * LIST INDEX VALUE
TSCAPTCH DS    CL22              * EXPANSION AREA

*************************************************************************
*
* THE FOLLOWING ARE CONSTANTS USED FOR THE ATTRIBUTE OF THE TSCA
* FIELDS. THEY ARE THE ONLY VALID SETTINGS FOR THE TSCA ATTR.
*
*************************************************************************
TSCACHAR EQU   X'01'              * ATTRIBUTE IS CHARACTER
TSCAFIXD EQU   X'02'              * ATTRIBUTE IS FIXED
TSCABIT  EQU   X'03'              * ATTRIBUTE IS BIT
TSCAPOIN EQU   X'04'              * ATTRIBUTE IS POINTER
TSCAACRY EQU   C'TSCA'            * TSCA ACRONYM
```

B

Assembler Code User Exit Example

The following user exit routine adds freeform text to a record when used with the TSP on page 51. This routine also adds one line of freeform text to a record that is already in update mode or create mode.

User Exit Example

```
TBLDATA  CSECT
         USING TBLDATA,R15
         STM   R14,R12,12(R13)        SAVE THE CALLERS REGISTERS
         B     START                  BRANCH AROUND EYECATCHER
         DC    CL8'TBLDATA '          EYECATCHER
START    EQU   *
         LR    R12,R15                USE OUR EPA FOR A BASE ADDRESS
         DROP  R15
         USING TBLDATA,R12            TELL ASSEMBLE OUR BASE REG
         L     R8,0(R1)               POINT REG1 TO THE TSCA
         USING TSCA,R8                TELL ASSEMBLE ABOUT THE TSCA
         ST    R13,SAVEAREA+4         PUT CALLER SAVE AREA IN OURS
         LA    R15,SAVEAREA           POINT TO OUR SAVE AREA
         ST    R15,8(R13)             TELL CALLER WHERE OUR SAVEAREA IS
         LR    R13,R15                SET UP OUR SA IN CASE NEEDED
*************************************************************************
*                      PLACE YOUR CODE BELOW*                           *
* REG13= OUR SAVE AREA     REG12= OUR BASE REG   REG8= PTR TO TSCA    *
*************************************************************************
         LA    R6,TSCATBLL            TARGET OF MVC
         MVC   0(20,R6),TEXTDATA      MOVE 20 BYTE FROM DATAONE TO REG6
         LA R6,TSCACTBL
         MVC 0(04,R6),TXLENGTH
*************************************************************************
*                      ALL DONE NOW                                     *
* SET RC0 INT REG15 AND TSCA.  THEN RESTORE REGS AND RETURN TO CALLER *
*************************************************************************
         LA    R15,0                  SIGNAL RETURN CODE FOR OPERATION
         ST    15,TSCAFRET            RETURN CODE = REG 15
         ST    15,TSCAFRES            REASON CODE = REG 15
         B     EXIT                   RETURN TO CALLER
EXIT     EQU   *
         L     R13,SAVEAREA+4         RESTORE SAVE AREA POINTER
         L     R14,12(R13)            RESTORE REGISTER 14
         LM    R0,R12,20(R13)         RESTORE CALLERS REGISTERS
         BR    R14
         EJECT
*************************************************************************
*                         CONSTANT AREA                                 *
*************************************************************************
         SPACE 1
SAVEAREA DC    18F'0'
TEXTDATA DC    C'TEXT ADDED VIA A TSP AND USEREXIT.'
TXLENGTH DC    X'00000022'
         SPACE 2
```

```
*********************************************************************
*                          REGISTER EQUATES                         *
*********************************************************************
         SPACE 1
R0       EQU   00
R1       EQU   01
R2       EQU   02
R3       EQU   03
R4       EQU   04
R5       EQU   05
R6       EQU   06
R7       EQU   07
R8       EQU   08
R9       EQU   09
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         SPACE 2
*********************************************************************
*                          TSCA CONTROL BLOCK                       *
*********************************************************************
CNTBLK   DS    C
         SPACE 2
         BLGUTSCA
         END
```

C

Relating Publications to Specific Tasks

Your data processing organization can have many different users performing many different tasks. The books in the Tivoli Information Management for z/OS library contain task-oriented scenarios to teach users how to perform the duties specific to their jobs.

The following table describes the typical tasks in a data processing organization and identifies the Tivoli Information Management for z/OS publication that supports those tasks. See "The Tivoli Information Management for z/OS Library" on page 311 for more information about each book.

Typical Tasks

Table 72. Relating Publications to Specific Tasks

If You Are:	And You Do This:	Read This:
Planning to Use Tivoli Information Management for z/OS	Identify the hardware and software requirements of Tivoli Information Management for z/OS. Identify the prerequisite and corequisite products. Plan and implement a test system.	*Tivoli Information Management for z/OS Planning and Installation Guide and Reference*
Installing Tivoli Information Management for z/OS	Install Tivoli Information Management for z/OS. Define and initialize data sets. Create session-parameters members.	*Tivoli Information Management for z/OS Planning and Installation Guide and Reference* *Tivoli Information Management for z/OS Integration Facility Guide*
	Define and create multiple Tivoli Information Management for z/OS BLX-SPs.	*Tivoli Information Management for z/OS Planning and Installation Guide and Reference*
	Define and create APPC transaction programs for clients.	*Tivoli Information Management for z/OS Client Installation and User's Guide*
	Define coupling facility structures for sysplex data sharing.	*Tivoli Information Management for z/OS Planning and Installation Guide and Reference*
Diagnosing problems	Diagnose problems encountered while using Tivoli Information Management for z/OS	*Tivoli Information Management for z/OS Diagnosis Guide*

Table 72. Relating Publications to Specific Tasks (continued)

If You Are:	And You Do This:	Read This:
Administering Tivoli Information Management for z/OS	Manage user profiles and passwords. Define and maintain privilege class records. Define and maintain rules records.	*Tivoli Information Management for z/OS Program Administration Guide and Reference* *Tivoli Information Management for z/OS Integration Facility Guide*
	Define and maintain USERS record. Define and maintain ALIAS record. Implement GUI interface. Define and maintain command aliases and authorizations.	*Tivoli Information Management for z/OS Program Administration Guide and Reference*
	Implement and administer Notification Management. Create user-defined line commands. Define logical database partitioning.	*Tivoli Information Management for z/OS Program Administration Guide and Reference*
	Create or modify GUI workstation applications that can interact with Tivoli Information Management for z/OS. Install the Tivoli Information Management for z/OS Desktop on user workstations.	*Tivoli Information Management for z/OS Desktop User's Guide*
Maintaining Tivoli Information Management for z/OS	Set up access to the data sets. Maintain the databases. Define and maintain privilege class records.	*Tivoli Information Management for z/OS Planning and Installation Guide and Reference* *Tivoli Information Management for z/OS Program Administration Guide and Reference*
	Define and maintain the BLX-SP. Run the utility programs.	*Tivoli Information Management for z/OS Operation and Maintenance Reference*
Programming applications	Use the application program interfaces.	*Tivoli Information Management for z/OS Application Program Interface Guide*
	Use the application program interfaces for Tivoli Information Management for z/OS clients.	*Tivoli Information Management for z/OS Client Installation and User's Guide*
	Create Web applications using or accessing Tivoli Information Management for z/OS data.	*Tivoli Information Management for z/OS World Wide Web Interface Guide*

Table 72. Relating Publications to Specific Tasks (continued)

If You Are:	And You Do This:	Read This:
Customizing Tivoli Information Management for z/OS	Design and implement a Change Management system. Design and implement a Configuration Management system. Design and implement a Problem Management system.	*Tivoli Information Management for z/OS Problem, Change, and Configuration Management*
	Design, create, and test terminal simulator panels or terminal simulator EXECs. Customize panels and panel flow.	*Tivoli Information Management for z/OS Terminal Simulator Guide and Reference* *Tivoli Information Management for z/OS Panel Modification Facility Guide*
	Design, create, and test Tivoli Information Management for z/OS formatted reports.	*Tivoli Information Management for z/OS Data Reporting User's Guide*
	Create a bridge between NetView® and Tivoli Information Management for z/OS applications. Integrate Tivoli Information Management for z/OS with Tivoli distributed products.	*Tivoli Information Management for z/OS Guide to Integrating with Tivoli Applications*
Assisting Users	Create, search, update, and close change, configuration, or problem records. Browse or print Change, Configuration, or Problem Management reports.	*Tivoli Information Management for z/OS Problem, Change, and Configuration Management*
	Use the Tivoli Information Management for z/OS Integration Facility.	*Tivoli Information Management for z/OS Integration Facility Guide*
Using Tivoli Information Management for z/OS	Learn about the Tivoli Information Management for z/OS panel types, record types, and commands. Change a user profile.	*Tivoli Information Management for z/OS User's Guide*
	Learn about Problem, Change, and Configuration Management records.	*Tivoli Information Management for z/OS Problem, Change, and Configuration Management*
	Receive and respond to Tivoli Information Management for z/OS messages.	*Tivoli Information Management for z/OS Messages and Codes*
	Design and create reports.	*Tivoli Information Management for z/OS Data Reporting User's Guide*

D

Tivoli Information Management for z/OS Courses

Education Offerings

Tivoli Information Management for z/OS classes are available in the United States and in the United Kingdom. For information about classes outside the U.S. and U.K., contact your local IBM representative or visit **http://www.training.ibm.com** on the World Wide Web.

United States

IBM Education classes can help your users and administrators learn how to get the most out of Tivoli Information Management for z/OS. IBM Education classes are offered in many locations in the United States and at your own company location.

For a current schedule of available classes or to enroll, call 1-800-IBM TEACh (1-800-426-8322). On the World Wide Web, visit:

http://www.training.ibm.com

to see the latest course offerings.

United Kingdom

In Europe, the following public courses are held in IBM's central London education centre at the South Bank at regular intervals. On-site courses can also be arranged.

For course schedules and to enroll, call Enrollments Administration on 0345 581329, or send an e-mail note to:

contact_educ_uk@vnet.ibm.com

On the World Wide Web, visit:

http://www.europe.ibm.com/education-uk

to see the latest course offerings.

E

Where to Find More Information

The Tivoli Information Management for z/OS library is an integral part of Tivoli Information Management for z/OS. The books are written with particular audiences in mind. Each book covers specific tasks.

The Tivoli Information Management for z/OS Library

The publications shipped automatically with each Tivoli Information Management for z/OS Version 7.1 licensed program are:

- *Tivoli Information Management for z/OS Application Program Interface Guide*
- *Tivoli Information Management for z/OS Client Installation and User's Guide **
- *Tivoli Information Management for z/OS Data Reporting User's Guide **
- *Tivoli Information Management for z/OS Desktop User's Guide*
- *Tivoli Information Management for z/OS Diagnosis Guide **
- *Tivoli Information Management for z/OS Guide to Integrating with Tivoli Applications **
- *Tivoli Information Management for z/OS Integration Facility Guide **
- *Tivoli Information Management for z/OS Licensed Program Specification*
- *Tivoli Information Management for z/OS Master Index, Glossary, and Bibliography*
- *Tivoli Information Management for z/OS Messages and Codes*
- *Tivoli Information Management for z/OS Operation and Maintenance Reference*
- *Tivoli Information Management for z/OS Panel Modification Facility Guide*
- *Tivoli Information Management for z/OS Planning and Installation Guide and Reference*
- *Tivoli Information Management for z/OS Program Administration Guide and Reference*
- *Tivoli Information Management for z/OS Problem, Change, and Configuration Management**
- *Tivoli Information Management for z/OS Reference Summary*
- *Tivoli Information Management for z/OS Terminal Simulator Guide and Reference*
- *Tivoli Information Management for z/OS User's Guide*
- *Tivoli Information Management for z/OS World Wide Web Interface Guide*

Note: Publications marked with an asterisk (*) are shipped in softcopy format only.

Also included is the Product Kit, which includes the complete online library on CD-ROM.

To order a set of publications, specify order number SBOF-7028-00.

Additional copies of these items are available for a fee.

Publications can be requested from your Tivoli or IBM representative or the branch office serving your location. Or, in the U.S., you can call the IBM Publications order line directly by dialing 1-800-879-2755.

The following descriptions summarize all the books in the Tivoli Information Management for z/OS library.

Tivoli Information Management for z/OS Application Program Interface Guide, SC31-8737-00, explains how to use the low-level API, the high-level API, and the REXX interface to the high-level API. This book is written for application and system programmers who write applications that use these program interfaces.

Tivoli Information Management for z/OS Client Installation and User's Guide, SC31-8738-00, describes and illustrates the setup and use of Tivoli Information Management for z/OS's remote clients. This book shows you how to use Tivoli Information Management for z/OS functions in the AIX®, CICS®, HP-UX, OS/2®, Sun Solaris, Windows NT®, and OS/390 UNIX System Services environments. Also included in this book is complete information about using the Tivoli Information Management for z/OS servers.

Tivoli Information Management for z/OS Data Reporting User's Guide, SC31-8739-00, describes various methods available to produce reports using Tivoli Information Management for z/OS data. It describes Tivoli Decision Support for Information Management (a Discovery Guide for Tivoli Decision Support), the Open Database Connectivity (ODBC) Driver for Tivoli Information Management for z/OS, and the Report Format Facility. A description of how to use the Report Format Facility to modify the standard reports provided with Tivoli Information Management for z/OS is provided. The book also illustrates the syntax of report format tables (RFTs) used to define the output from the Tivoli Information Management for z/OS REPORT and PRINT commands. It also includes several examples of modified RFTs.

Tivoli Information Management for z/OS Desktop User's Guide, SC31-8740-00, describes how to install and use the sample application provided with the Tivoli Information Management for z/OS Desktop. The Tivoli Information Management for z/OS Desktop is a Java-based graphical user interface for Tivoli Information Management for z/OS. Information on how to set up data model records to support the interface and instructions on using the Desktop Toolkit to develop your own Desktop application are also provided.

Tivoli Information Management for z/OS Diagnosis Guide, GC31-8741-00, explains how to identify a problem, analyze its symptoms, and resolve it. This book includes tools and information that are helpful in solving problems you might encounter when you use Tivoli Information Management for z/OS.

Tivoli Information Management for z/OS Guide to Integrating with Tivoli Applications, SC31-8744-00, describes the steps to follow to make an automatic connection between NetView and Tivoli Information Management for z/OS applications. It also explains how to customize the application interface which serves as an application enabler for the NetView Bridge and discusses the Tivoli Information Management for z/OS NetView AutoBridge. Information on interfacing Tivoli Information Management for z/OS with other Tivoli management software products or components is provided for Tivoli Enterprise Console, Tivoli Global Enterprise Manager, Tivoli Inventory, Tivoli Problem Management, Tivoli Software Distribution, and Problem Service.

Tivoli Information Management for z/OS Integration Facility Guide, SC31-8745-00, explains the concepts and structure of the Integration Facility. The Integration Facility provides a task-oriented interface to Tivoli Information Management for z/OS that makes the

Tivoli Information Management for z/OS applications easier to use. This book also explains how to use the panels and panel flows in your change and problem management system.

Tivoli Information Management for z/OS Master Index, Glossary, and Bibliography, SC31-8747-00, combines the indexes from each hardcopy book in the Tivoli Information Management for z/OS library for Version 7.1. Also included is a complete glossary and bibliography for the product.

Tivoli Information Management for z/OS Messages and Codes, GC31-8748-00, contains the messages and completion codes issued by the various Tivoli Information Management for z/OS applications. Each entry includes an explanation of the message or code and recommends actions for users and system programmers.

Tivoli Information Management for z/OS Operation and Maintenance Reference, SC31-8749-00, describes and illustrates the BLX-SP commands for use by the operator. It describes the utilities for defining and maintaining data sets required for using the Tivoli Information Management for z/OS licensed program, Version 7.1.

Tivoli Information Management for z/OS Panel Modification Facility Guide, SC31-8750-00, gives detailed instructions for creating and modifying Tivoli Information Management for z/OS panels. It provides detailed checklists for the common panel modification tasks, and it provides reference information useful to those who design and modify panels.

Tivoli Information Management for z/OS Planning and Installation Guide and Reference, GC31-8751-00, describes the tasks required for installing Tivoli Information Management for z/OS. This book provides an overview of the functions and optional features of Tivoli Information Management for z/OS to help you plan for installation. It also describes the tasks necessary to install, migrate, tailor, and start Tivoli Information Management for z/OS.

Tivoli Information Management for z/OS Problem, Change, and Configuration Management, SC31-8752-00, helps you learn how to use Problem, Change, and Configuration Management through a series of training exercises. After you finish the exercises in this book, you should be ready to use other books in the library that apply more directly to the programs you use and the tasks you perform every day.

Tivoli Information Management for z/OS Program Administration Guide and Reference, SC31-8753-00, provides detailed information about Tivoli Information Management for z/OS program administration tasks, such as defining user profiles and privilege classes and enabling the GUI user interface.

Tivoli Information Management for z/OS Reference Summary, SC31-8754-00, is a reference booklet containing Tivoli Information Management for z/OS commands, a list of p-words and s-words, summary information for PMF, and other information you need when you use Tivoli Information Management for z/OS.

Tivoli Information Management for z/OS Terminal Simulator Guide and Reference, SC31-8755-00, explains how to use terminal simulator panels (TSPs) and EXECs (TSXs) that let you simulate an entire interactive session with a Tivoli Information Management for z/OS program. This book gives instructions for designing, building, and testing TSPs and TSXs, followed by information on the different ways you can use TSPs and TSXs.

Tivoli Information Management for z/OS User's Guide, SC31-8756-00, provides a general introduction to Tivoli Information Management for z/OS and databases. This book has a series of step-by-step exercises to show beginning users how to copy, update, print, create, and delete records, and how to search a database. It also contains Tivoli Information Management for z/OS command syntax and descriptions and other reference information.

Tivoli Information Management for z/OS World Wide Web Interface Guide, SC31-8757-00, explains how to install and operate the features available with Tivoli Information Management for z/OS that enable you to access a Tivoli Information Management for z/OS database using a Web browser as a client.

Other related publications include the following:

Tivoli Decision Support: Using the Information Management Guide is an online book (in portable document format) that can be viewed with the Adobe Acrobat Reader. This book is provided with Tivoli Decision Support for Information Management (5697-IMG), which is a product that enables you to use Tivoli Information Management for z/OS data with Tivoli Decision Support. This book describes the views and reports provided with the Information Management Guide.

IBM Redbooks™ published by IBM's International Technical Support Organization are also available. For a list of redbooks related to Tivoli Information Management for z/OS and access to online redbooks, visit Web site **http://www.redbooks.ibm.com** or **http://www.support.tivoli.com**

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

File Number: S370/30xx/4300
Program Number: 5697-SD9

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC31-8755-00