CA Telon® Application Generator

Test Facility Guide

r5.1

CA Product References

This document references the following CA products:

- CA Telon® Application Generator (CA Telon)
- CA Datacom

Contact CA

Contact Technical Support

For your convenience, CA provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA products. At http://ca.com/support, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Provide Feedback

If you have comments or questions about CA product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA product documentation, complete our short customer survey, which is also available on the CA Support website, found at http://ca.com/docs.

Contents

Chapter 1: Introduction

The online CA Telon Test Facility (TTF) is an interactive facility that allows you to test your completed COBOL or PL/I programs in a hands-on environment. The TTF allows you to simulate real-time usage while operating in an independent, controlled environment on the mainframe.

Concepts and Methodology

This guide discusses procedures using the CA Telon Test Facility (TTF) for testing and prototyping your completed COBOL or PL/I programs in a hands-on environment throughout the application development cycle. In this manual, Advantage CA Telon Application Generator, formerly known as CA Telon, is referred to simply as CA Telon.

Audience

This manual is intended for application programmers responsible for testing programs developed with CA Telon.

Where to Find Additional Information

You can find additional information about CA Telon in the guides that comprise the CA Telon documentation set. The README.TXT contained on the documentation CD provides a list of these guides.

Methods of Testing

This chapter covers various methods for testing CA Telon-generated applications. The different development and target environments have special requirements and/or options for testing. No matter what the development or target environment, testing of programs is strongly recommended.

PWS Testing

CA Telon Programmable Work Station (PWS) does not include a test facility because quality debuggers are available for PC applications.

Mainframe Testing

The CA Telon Test Facility is the most convenient way to test mainframe applications because it is easy to access. Before testing, you must use the CA Telon Generator to generate your program into structured COBOL, or PL/I code. See the *Programming Concepts Guide* for more information on generating programs.

This generated, native code contains no information specific to CA Telon. Since CA Telon generates stand-alone programs, you can actually test them with *any* online testing product.

The TTF retains the distinct advantage of allowing you to trace database and file access calls and to interrogate storage at any desired point. TTF debug features are especially helpful for applications developed on CA Telon. For example, the TTF can stop execution before and/or after accessing files or databases. You can also test field edits. At these stop points, you can make sure that the calls are correct. The CA Telon Test Facility also allows you to prototype applications with a greater degree of accuracy than with the prototyping facility. The TTF lets you test individual programs, even if dependent programs do not yet exist.

As a part of the whole CA Telon development system, the TTF ties in with the other components so you can design, prototype, generate and test your applications within the same product, CA Telon. The TTF integrates with the rest of CA Telon to provide a broad approach to application development. At every stage in the development process, CA Telon provides methods to prototype and test what you have developed.

Design Facility Output

The CA Telon Design Facility is an online tool that you can use to develop individual programs and entire application systems. It produces CA Telon source statements that are input to the CA Telon Application Generator to create COBOL or PL/I source code. Each element of a CA Telon program produced in the TDF is explained below.

Panel Image

The panel image identifies the format of the screen. You create a panel image by **painting** it on the screen exactly as it should appear to the end-user. CA Telon captures literals and their location from the panel image. Literals are unchangeable fields that supply information to the application user during program execution. Literals can be field or group headings or instructional messages that always appear on a screen. CA Telon captures the type, length, and location of variable fields by means of special symbols that you paint on the panel image.

Panel Definition

The panel definition builds on the panel image by specifying the processing characteristics of each field, such as the source of displayed data, the destination of the entered data, and the field attributes. You can also specify edit criteria, data cross validation, and key validation.

Program Definitions

The program definitions build on the panel definition to supply general program characteristics that include cursor positioning, screen flow, data access, PF key processing, and custom COBOL or PL/I code. You can create several types of program definitions in the TDF, which include the screen definition or SD, batch definition or BD, non-terminal definition or ND, and report definition or RD. See the *Design Facility Reference* and *Programming Concepts Guide* for more information on the panel image, panel definition, and various program definitions.

Generator Output

The CA Telon Generator is a batch utility that produces native COBOL or PL/I source code either for application testing or production execution.

CA Telon source statements from the design facility serve as input to the Generator. The Generator's output is a stand-alone, native COBOL or PL/I source program. The source program contains all necessary processing control information.

CA Telon-generated programs:

- Conform to a standard architectural structure
- Contain your custom code COBOL or PL/I statements just as you coded them

The standard structure provides high development and maintenance productivity without sacrificing performance.

Since CA Telon generates a native COBOL or PL/I source program, no CA Telon run-time monitor or other **black box** is necessary for program compilation or execution. This means that you can use any online testing product to test your CA Telon generated programs.

Test Facility Output

The CA Telon Test Facility provides an interactive debugging tool that allows you to test an application or prototype new screens that you created with either CA Telon Batch or Online.

The TTF allows you to execute any CA Telon Screen Definition without following the logical processing sequence. That is, you do not have to execute, or even create, the screens that precede the screen you are testing. This allows you to test an individual module even before you create the preceding modules or any modules that follow. CA Telon simply halts execution if control is supposed to pass to a program that CA Telon does not recognize.

Full-Screen Interactive Debugging

The TTF provides screen-to-screen control and full-screen interactive debugging facilities. Among its other features and functions, it allows you to:

- Display program information, such as program name and compilation date and time
- Display and modify program and working storage areas, in both character and hexadecimal format
- Stop execution at specified instructions or break points
- Intercept and interactively debug program ABENDs

Tracing

In addition to helping you debug data-related problems, the TTF gives you a dynamic, full-screen tracing of every DL/I database call, DB2, IDMS SQL, and DATACOM table access call, and VSAM, or sequential file access. You can request a trace screen to indicate the status before and/or after each call. You can set or reset the trace parameters at any time during the testing process. The TTF controls the loading and execution of CA Telon screen programs. The online CA Telon Test Facility runs in TSO for testing IMS or Batch programs.

Methods of Prototyping and Testing

Just as CA Telon provides components to perform every stage of the development process, it also provides opportunities for prototyping and testing at every stage of the application development process. As you complete each step in the development process, CA Telon provides methods that allow you to try the application before continuing.

Depending on how much of the application you have completed, you can:

- Prototype an application for demonstration to end users
- Test the generated code for individual screens

At any stage of application development, you can always return to an earlier level of prototyping. Even if you have already created a screen definition, you can use a method that only requires the panel image. At any stage of development you can consider the application, even if it is already in production, to be the current prototype. After initial testing, it is easy to return to the TDF and make changes. The final step in prototyping is testing with the TTF.

CA Telon provides five different types of prototyping. In order from least to greatest sophistication, or in similarity to eventual production systems, these types of prototyping are:

- Prototyping without data mapping
- Prototyping with data mapping
- Screen execution without data access
- Screen execution with generic data access
- Screen execution with real/production data access

In the CA Telon Design Facility

The first two types of prototyping, prototyping without and with data mapping, are available through the prototyping facility in the TDF.

Prototyping Without Data Mapping

You can create application scenarios very quickly with this level of prototyping. The only required design component is a *panel image*. You can present screen designs to the application user for review. The user can comment on screen fields, functions, and screen-to-screen flow. You can enter data in the screens and use special commands to move from screen to screen to simulate proposed work flow, but the data is not stored for later display.

Prototyping With Data Mapping

You can create more realistic application scenarios with this level, which requires the *panel definition*. You can present screen designs to the application user for review, both for general screen content and for screen-to-screen flow. You can enter data, edit it, and transfer it to later screens in the scenario. Data is stored in a Presentation Store, the means by which you simulate database access.

In the CA Telon Test Facility

The remaining types of prototyping require the CA Telon Test Facility and a program definition. Prototypes created under the TTF can be extremely realistic. Each level of simulation, described in detail below, builds on the previous level, but you can always return to a lower level. As more functions become available, the simulations become more useful. To implement any level of prototyping within the CA Telon Test Facility, generate a program from TDF input and execute it under the Test Facility.

Screen Execution without Data Access

This level lets you show field and consistency editing, as well as the automatic flow of control from screen to screen. In addition, you can illustrate PF Key processing, application algorithms, and functions such as help and hold.

Screen Execution with Generic Data Access

This level adds more realistic simulation of data access, when the databases for the application have yet to be completed. Generic databases can provide test data and add a level of realism that makes the prototype very close to the production application.

Screen Execution With Data Access

This is similar to prototyping with generic data access, but you replace the generic databases with the real application databases. The application execution does not appear differently to the application user. You can analyze the final database structure with the application process as it appears to the application user.

The table below shows the stages of CA Telon-developed applications and the types of testing that you can use:

Completed	Available Methods at this Stage
Panel image	Prototyping without data mapping (TDF)
Panel definition	Prototyping with data mapping (TDF)
Screen definition	Screen execution (TTF): ■ Without data access ■ With generic data access ■ With application databases

These methods are explained at greater length elsewhere in this guide. See the *Programming Concepts Guide* for more information on the prototyping facility.

Chapter 2: Screen Descriptions

This chapter provides the information necessary to use the CA Telon Test Facility on the mainframe. It discusses the TTF's organization, screens, and fields. Specifically, this chapter contains:

- **Test Facility Screen Organization Chart**, including a brief description of the TTF screens and their function
- **Screen descriptions** explaining the following TTF screens and their fields:
  - Field Edit Test menu
  - Update OS Parameter screen
  - Debug Application screen
  - Trace DL/I Access screen
  - Trace DB2 Access screen
  - Update DL/I Trace screen
  - Update DB2 Trace screen
  - Display Full Screen Storage screen
  - List Module Map screen
  - List TSO Loaded Modules screen

Using the Test Facility

The CA Telon Test Facility is an interactive tool that you can use to test CA Telon COBOL and PL/I:

- TSO and IMS programs
- Batch programs under TSO

You can use the CA Telon Test Facility for both full-systems testing and prototyping, that is, testing of partially completed systems. The Test Facility provides screen-to-screen control, interactive debugging, and dynamic file access tracing.

Screen-to-Screen Control

The TTF allows you to execute any CA Telon generated module independent of the modules on which it depends. During testing, all program execution is controlled by a single TSO CLIST. The CA Telon Test Facility Main Menu begins each testing session and passes control to the requested program, regaining control if control is passed to a non-existent program or if an ABEND occurs.

Interactive Debugging

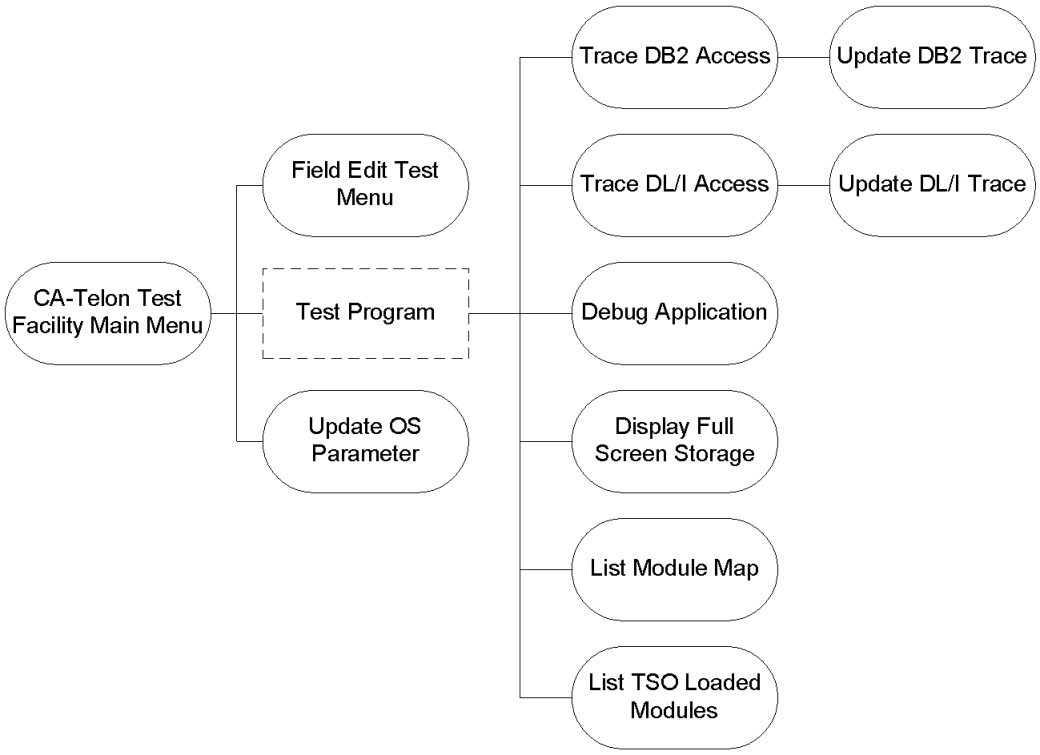
The full-screen interactive debug feature assists you in the testing of application programs. It allows for display and update of storage, stopping at specified breakpoints during program execution, altering the flow of screens, and ABEND or abnormal termination analysis.

Dynamic File Access Tracing

The TTF provides, upon request, a dynamic, full-screen access trace of every DL/I database call, DB2, IDMS SQL, or DATACOM table access call, VSAM, and sequential file access. The trace occurs before and/or after the file access. You can choose to bypass the data access after you see the before call trace screen.

Test Facility Screen Organization Chart Illustration

The organization chart for CA Telon Test Facility screens is illustrated below.



Test Facility Screen Organization Chart

The Test Facility Organization Chart consists of the following test program and screens:

Program/Screen	Description
Application Test Program	Consists of any programs compiled for execution and testing in the TSO environment.
CA Telon Test Facility Main Menu	The entry and exit point of the Test Facility. It allows you to select the test program name, set up test options and file trace options, and update system information that the test application uses.
Field Edit Test Menu	The primary entry point for testing CA Telon-supplied and user-written field edits. It allows you to select the edit program name that you want to test and to select parameters that you want to pass to the edit.
Update OS	Allows you to view and update the value of the OS

Program/Screen	Description
Parameter Screen	Parameter that you want to pass to the test program.
Debug Application	The main focal point from which you perform application debugging. It allows you to view and update storage, set and remove breakpoints, update trace variables, and continue or restart execution of the test application.
Trace DL/I Access Screen	Allows you to view DL/I database access before and after the access takes place.
Trace DB2 Access Screen	Allows you to view DB2 table access before and after the access takes place.
Update DL/I Trace Screen	Allows you to view the DL/I PCBs and select before and after-tracing of DL/I access on the PSB and PCB level. It also allows restricting the trace to certain DL/I and DB2 functions.
Update DB2 Trace Screen	The Update DB2 Trace screen allows you to select before and after tracing of DB2 access by DB2 SQL commands.
Display Full Screen Storage Screen	Allows you to view and update storage. The TTF displays the storage in a full screen format. It is in effect an extension of the smaller storage display area on the Debug Application screen.
List Module Map Screen	Allows you to view the link-edit map of the test program. In the TSO environment, this screen also displays the link-edit map of any module loaded in the Test Facility address space.
List TSO Loaded Modules Screen	Allows you to view a list of modules loaded in the user's address space.

Invoking and Running the Test Facility

Follow the steps outlined below to prepare for and use the CA Telon Test Facility. For CA Telon Test Facility installation considerations, see the "Installation Considerations" appendix.

Request Trace Tables

To generate the trace logic that the TTF requires, you must request program tracing from one of the TDF Environment screens. Select one of the following options on the CA Telon Design Facility Main Menu to access the appropriate Update Environment screen:

- **1** - User profile maintenance. Choose **D**, DEFINITION DEFAULTS. Select UPDATE ENVIRON DEFN DEFAULTS. This screen allows you to request program tracing for IMS and Batch programs. Enter **Y** in the TRACE field for your particular environment.
- **4** - Online program definition. Update ITEM EN, the environment. Enter **Y** in the TRACE field of the Update Environment screen.
- **5** - Batch program definition. Update ITEM EN, the environment. Enter **Y** in the TRACE field of the Update Environment screen.

Note: You also can update the environment while you update the entire screen, nonterminal, or batch definition. The OPTIONS line on the Update Program Definition screen contains one of the following choices: ENV IMS, ENV TSO, ENV BATCH, or ENV AS400. Select the available option to access the appropriate Update Environment screen. Enter **Y** in the TRACE field on that screen.

Allocating Data

Before starting the CA Telon Test Facility, you must properly allocate the data according to the database your programs use. Depending on the method of access, you might need to establish DB2 Plans, Batch Output Files, and/or available PSBs.

Invoking the Test Facility

The CA Telon Test Facility tests TELON TSO, IMS, and Batch programs. It includes an option to perform field edit, or subroutine, testing.

According to your CA Telon environment, execute the appropriate transaction. Your CA Telon Coordinator can provide the name of the CLIST to invoke. When the TTF gets control, it displays the CA Telon Test Facility Main Menu.

Note: For information on Telon-supplied CLISTS for data allocation and invoking the Test Facility, see the Test Facility CLISTs section in the "Installation Considerations" appendix.

Entering the Program Name

The cursor is now positioned at the TEST PROGRAM NAME field on the Test Facility Main Menu. Enter the name of the first program you want executed. The name you enter here is the name of the load module for the application you are testing.

You do not have to begin with the program that will be executed first during actual production runs of the application. This way, you can test a program before you complete the program that logically precedes it. If the test program requires values in the Transfer Work Area, you must set up these values either with another program, or manually on the Debug Application screen.

Changing Trace Option Settings (optional)

If desired, you can alter the Trace Option settings on the Test Facility Main Menu before you begin testing. These options are described in this chapter under the CA Telon Test Facility Main Menu. If you set TRACE BEFORE to **Y**, the TTF displays a Trace Access Before screen preceding any data access. If you set TRACE AFTER to **Y**, the TTF displays a Trace Access After screen following any data access.

Setting Refresh and Trace

The TTF provides you with better debugging information if you specify **Y** in the TRACE, PGMINIT and REFRESH working storage fields on the Test Facility Main Menu.

Beginning Testing

Press Enter or the TEST GO key. If CA Telon cannot find the specified program, an error message appears. Enter a corrected program name. If CA Telon finds the specified program, the TTF begins executing it.

As the Test Facility proceeds, it returns Debug screens, DL/I and DB2 Trace screens, and File Trace screens according to the Trace Options entered on the CA Telon Test Facility Main Menu. The TTF also intercepts ABENDs.

Transfering Control to Another Program

When the executing program requests that control pass to another program, the TTF performs the transfer.

If the next program does not exist, as often happens during prototyping, the TTF returns to the CA Telon Test Facility Main Menu. The Main Menu displays the program name and the message PROGRAM DOES NOT EXIST.

At this point, you can perform one of the following activities:

- Enter the name of a program that does exist to test
- Terminate testing

Each of these activities is discussed below.

Terminating the Test Facility

To terminate the TTF and exit to TSO, you can:

- Type **=X** on the COMMAND field of any Test Facility screen
- Type **END** or **=X**. in the TEST PROGRAM NAME field of the CA Telon Test Facility Main Menu
- Press PF3 until you reach the desired point

Test Facility Main Menu

To begin program testing on the CA Telon Test Facility Main Menu, enter the name of the first test program to execute in the TEST PROGRAM NAME field. The TRACE OPTIONS below this field allow you to request other debugging and trace facilities that the TTF offers. If you operate with all the trace options off, that is, the TRACE OPTIONS field value is **N** (no), program execution resembles native IMS execution; the TTF does not return trace or debugging screens to you.

An example of the CA Telon Test Facility Menu is shown below.

```
TELON TEST FACILITY MAIN MENU ******* ****************************************
COMMAND ==> 1_____________________________________________________________________
OPTIONS ==> TEST FIELD EDIT 2      UPDATE PARAMETER LIST 3

                                   PARAMETER LIST FORMAT DLI 4 (OS,DLI)

 TEST PROGRAM NAME 5_______

 ----------------------- TRACE OPTIONS ---------------------------
                                                          BEFORE  AFTER
 STOP AT PROGRAM INIT 6 N       FILE ACCESS 7          N       N



 MISCELLANEOUS OPTIONS:                      SYSTEM INFORMATION:
 ----------------------------------      ---------------------------------
 TEST AID KEY PF12  8                       LTERM  12 SMITH___     DATE  11/03/01
 TEST GO KEYS PF10 PF22  9                  USERID 13 SMITH___    TIME  11:28:32
 TSO TRANSFER WORK AREA SIZE   32 K 10
 REFRESH WORKING STORAGE Y 11
```

Fields

1	COMMAND	This field allows you to enter any of the valid test commands activated for the current test screen. See the "Commands" appendix for a complete list of TTF commands.
2	TEST FIELD EDIT	The Test Field Edit option transfers control to the Field Edit Test Menu for execution of CA Telon-supplied or user-written field edits. Enter any non-blank character in this field to transfer to the Field Edit Test Menu.
3	UPDATE PARAMETER LIST	This option transfers control to either the Update OS Parameter screen or the Update DL/I Trace screen depending on the value of the PARAMETER LIST FORMAT field. Enter any non-blank character in this field to transfer control to the appropriate screen.
4	PARAMETER	This field allows you to select the type of parameters

	LIST FORMAT	passed to the program being tested. Values for this field can be OS and DL/I. If DL/I is not active in invoking the Test Facility, CA Telon sets from user update.
5	TEST PROGRAM NAME	This field is required for testing to begin. You must enter a valid test program name in this field. Then press either the Enter or the TEST GO key to begin test program execution.
6	STOP AT PROGRAM INIT	This field allows you to request the TTF to return the Debug Application after each new test program is loaded (and before it executes). Valid values are: ■ **Y** (yes) ■ **N** (no) Use this option to view and possibly modify initial values (especially in the Transfer Work Area) that the test program expects.
7	FILE ACCESS	This field allows you to request CA Telon to display the trace screen appropriate to the access either before or after the I/O. For each of the headings, BEFORE and AFTER, enter **Y** (yes) to request CA Telon to display the trace screen appropriate to the access either before or after the I/O. CA Telon displays the trace screen for each attempted access. Enter **N** (no) if you do not want to see the trace screens. The CA Telon Test Facility intercepts the test program calls to the ABEND modules ADGADBER and ADGDBER. The TTF then displays the appropriate file access trace screen with the message TRACE CALLED BY ABEND ROUTINE. In this case, the settings of the general FILE ACCESS indicators remain unchanged. For DL/I databases, the trace display shows the appropriate file access trace screen; for DB2 tables, the trace display shows the Trace DB2 Access screen. If you enter **Y** (yes) in these fields, you can override these values at a lower level on the Update DL/I Trace screen and Update DB2 Trace screen. See the appropriate screen descriptions (for example, "Trace DL/I Access Screen") in this chapter for more information.

8	TEST AID KEY	Enter the value of a PF Key that you want to use to return to the Debug Application screen when the test program has written its screen to the terminal. You can use any PF key here. During testing, when you press the defined key from an application screen, CA Telon transfers you to the Debug Application screen. **Note:** The TTF intercepts the TEST AID key and therefore, the test program cannot process it. You can also use the TEST AID key to transfer you to the Debug Application screen from any other screen in the TTF. When processing from the Debug Application screen, the TEST AID key issues the GO command to return to the test program or the appropriate Access Trace screen, depending where you initially invoked the Debug Application screen. If you set the TEST AID key to a value equal to one of the TEST GO keys, the TEST AID key takes precedence over the TEST GO key. Additionally, if these two keys are the same, Enter acts as a GO command from the Access Trace screens for compatibility with pre-2.0 releases of the TTF. You should not have a TEST AID key value equal to one of the TEST GO keys values. The default TEST AID key value is defined at installation time in the Test System Definition Module. The delivered default TEST AID key value is [PF12]
9	TEST GO KEYS	Enter values for the PF keys that cause the continuation of test program execution from any of the TTF screens. Valid values in these fields are PF*nn*, where *nn* is a PF key number not already assigned as a TTF command. See the "Installation Considerations" appendix in this guide for a list of keys already assigned as TTF commands. Note that if the TEST AID key is equal to one of the TEST GO keys, the TEST AID key takes precedence in processing. See the TEST AID key explanation for more information. The default TEST GO key values are defined at installation time. The delivered defaults are [PF10] and [PF22].
10	TSO TRANSFER WORK AREA SIZE	This field allows you to specify (in 1K-byte increments) the Transfer Work Area to be allocated for TSO test applications. The valid values for this field are the numbers 3 to 1000 (3K to 1 MEG). The default value for this field is defined at installation time. The delivered default is 32K.

11	REFRESH WORKING STORAGE	This field allows you to specify whether to refresh all areas of the TSO test program's working storage. This refresh occurs after the TTF displays the test program's screen. The TTF does not refresh the TP buffers. Testing with this value set to Y helps ensure that transfer data is not kept in a program's working storage across screen iterations. It also helps avoid application errors after conversion to IMS. The default value for this field is defined at installation time. Valid values are: ■ Y -(Default) Yes ■ N -No
12	LTERM	This field allows you to specify a logical terminal name for use in application testing. The TTF places the value of this field in the IMS I/O PCB (if available). The default value for this field is the TSO user ID.
13	USERID	This field allows you to specify a user ID for use in application testing. The TTF places the value of this field in the IMS I/O PCB (if available). The default value for the field is the TSO user ID.

Field Edit Test Menu

The Field Edit Test Menu receives control from the CA Telon Test Facility Main Menu when you select the TEST FIELD EDIT option. Use the Field Edit Test Menu to test both CA Telon-supplied and user-written input and output field edits. This screen allows you to specify the standard variables passed to field edits, as well as one to nine extension parameters to be appended to the standard parameter list. An example of the Field Edit Test Menu is shown below.

```
FIELD EDIT TEST MENU ********** ****************************************
COMMAND ==> 1_____________________________________________________________

 FIELD EDIT NAME 2_____     EDIT TYPE 3 (A/N/B/V)   WORKFLD NUMERIC 4_ V __

 STOP AT PROGRAM INIT 5_

 --------------------------- STANDARD PARAMETERS ----------------------------
 LENGTH: _6_ (TPO/TPI)
         ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
 INPUT:  7_____________________________________________________________________
   *      _____________________________________________________________________
 OUTPUT: 8
   *
 ERROR:  9

 --------------------------- EXTENDED PARAMETERS -----------------    PAGE 01
 NBR TYPE                          VALUE/DBDNAME/PCB NUMBER
NUMERIC
  1  _11 ___12_____________________________________________          13 V __
  2  ___ _____________________________________________________    13 V __
  3  ___ _____________________________________________________    13 V __
  4  ___ _____________________________________________________    13 V __
  5  ___ _____________________________________________________    13 V __
 10
```

There are two types of field edit; input and output. Input field edits must have a program name beginning with the letter I. They are passed the following standard parameters:

Field Edit Error Flag	4 bytes
TPI Field Length	Halfword binary
TPI Field	Screen Buffer Field
WORKFLD-ALPHA or	256-byte character
WORKFLD-NUMERIC	Display numeric based on default language specification or overridden value specified on screen
WORKFLD-BIT or	32-byte bit string
WORKFLD-VARCHAR	256-byte character string with halfword binary length field prefix

Output field edits must have a program name beginning with the letter O. They are passed the following standard parameters:

TPO Field	Screen Buffer Field
TPO Field Length	Halfword binary
WORKFLD-ALPHA or	256-byte character
WORKFLD-NUMERIC or	Display numeric based on default language specification or overridden value specified on screen
WORKFLD-BIT or	32-byte bit string
WORKFLD-VARCHAR	256-byte character string with halfword binary length field prefix

Fields

The following describes the function and usage of each field. The numbers in the fields correspond to the numbers below.

1	COMMAND	This field allows you to enter any of the valid test commands activated for the current test screen. See the "Commands" appendix for a complete list of commands and more information.
2	FIELD EDIT NAME	This field is required for testing to begin. Enter an edit program name beginning with the letter I or the letter O into this field. Then press Enter or the TEST GO key to begin the test program.
3	EDIT TYPE	This field allows you to define the type of WORKFLD that you want to pass to the field edit to be tested. Valid types are: ■ **A** - (Default) Alpha ■ **N** - Numeric ■ **B** - Bit ■ **V** - Varchar ■ **L** - Lowercase alpha The TTF processes Edit Type L exactly like Edit Type A, except that the INPUT field (parameter 7) and the EDIT EXTENSION VALUE field(s) (parameter 12) are not translated automatically into uppercase.
4	WORKFLD	This field allows you to specify an override value for the

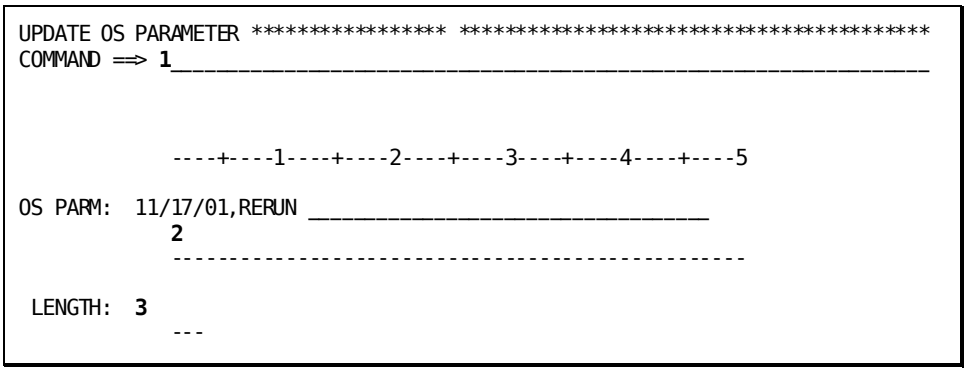
	NUMERIC	default WORKFLD-NUMERIC picture that the TTF passes to numeric type field edits. If you do not specify a value for WORKFLD NUMERIC, the TTF passes the following default values to the field edit: ■ S9(11)V9(7) - For COBOL ■ (10)9V4(9)T - For PL/I
5	STOP AT PROGRAM INIT	This field allows you to request the Test Facility to return the Debug Application screen after loading the test program (and before it executes). Values are: ■ **Y** -Yes ■ **N** -No
6	LENGTH	This field is required and allows you to specify the length of the screen field that the TTF passes to the edit. This field represents the length of the field as displayed on or keyed into an application screen. For input field edits it is the TPI-1th field, and for output field edits it is the TPO-1th field. For output field edits, this length does not necessarily represent the data keyed into the INPUT field.
7	INPUT	This field is required and specifies the data that the TTF passes as either screen input for input field edits or the data that the Test Facility converts to the correct WORKFLD format and then passes to output field edits for display. The WORKFLD passed (ALPHA, NUMERIC, BIT, or VARCHAR) depends on the value that you specify in the EDIT TYPE field, discussed earlier in this section. When applicable, you can define extended parameters for any type of edit. For VARCHAR-type OUTPUT edits, the length field of the WORKFLD-VARCHAR may be set by specifying a length at the beginning of the input field enclosed in parentheses (such as (80) input data). If a field is not supplied, the length will be calculated from the INPUT DATA supplied.
8	OUTPUT	This field displays the edit results after execution of the Field edit. For input field edits, this field displays the specified workfld that the TTF passes to the input field edit. For output field edits, this field displays the data as re-formatted for display on an application screen. The WORKFLD passed (ALPHA, NUMERIC, BIT, or VARCHAR) depends on the value that you specify in the EDIT TYPE field, discussed earlier in this section. When applicable, you can define extended parameters for any type of edit.
9	ERROR	This field displays the four-byte FIELD-EDIT-ERROR field

		that the input field edits returns.
10	EDIT EXTENSION NBR	This field defines the relative number of the edit extension parameter that the TTF passes to the field edit. You can specify a maximum of nine edit extensions in these fields. Use the scroll keys to view extensions 6 through 9. Before execution of the edit, the TTF removes from the calling parameter list, any extension with no TYPE or VALUE specification. The TTF moves up all subsequent values as required.
11	EDIT EXTENSION TYPE	The EDIT EXTENSION TYPE field allows you to specify data typing of edit extension parameters. The valid types for this field are: ■ **CHR** for 256-byte character ■ **NUM** for display numeric ■ **DEC** for decimal (COMP-3 or FIXED DEC) ■ **H** for halfword binary (S9(4) COMP or FIXED BIN(15)) ■ **F** for fullword binary (S9(8) COMP or FIXED BIN(31)) ■ **BIT** for BIT ■ **XFR** for passing the Transfer Work Area as a parameter ■ **PCB** for passing a DL/I PCB ■ **VCH** for character varying If you do not specify a value for EDIT EXTENSION TYPE, the TTF uses a default value of CHR.
12	EDIT EXTENSION VALUE	The EDIT EXTENSION VALUE field defines the data that the TTF passes as an extension to the edit. The TTF converts the value that you input to the correct data type. It then passes the input value to the field edit. For BIT type fields, the size of the field that the TTF passes will be 32 bytes, padded on the right with binary zeros. For PCB type fields, the value that you enter for this field must be either a DBDNAME in the PSB or a number corresponding to the relative number of the PCB that the TTF passes. For VCH-type fields, the extension passed is a 256-byte character string with a two-byte binary-length header. The length field will be set to the length of the input data unless a **LTH** override is specified by keying the length value at the beginning of the field enclosed in parentheses (ex: (80) input data).
13	EDIT EXTENSION NUMERIC	The EDIT EXTENSION NUMERIC field allows you to specify an override value for the default numeric picture for numeric, decimal, and binary data types. The values that

	you specify represent the number of digits that the TTF passes to the left and to the right of the decimal point (represented by the V). If you do not specify a value for this parameter, the TTF passes the following default values to the field edit: ■ For numeric types - The default values are S9(11)V9(7) for COBOL and (10)9V4(9)T for PL/I ■ For binary types (H and F) -The default values passed are integer values not exceeding the limit of the binary field's size (32,767 for halfword and 2,147,483,647 for fullword)

Update OS Parameter Screen

The Update OS Parameter screen receives control after you enter the DISPLAY OSPARM command, described later in this chapter, or if you enter the UPDATE PARAMETER LIST option on the CA Telon Test Facility Main Menu. Use this screen to specify an OS Parameter that you want to pass to the test program.



Fields

The following describes the function and usage of each field. The numbers in the fields correspond to the numbers below.

1	COMMAND	This field allows you to enter any of the valid test commands activated for the current test screen. See the "Commands" appendix for a complete list of and more information.
2	OS PARM	This field is the value of the OS parameter that you want to pass to the test program. The value entered into this field is passed to the test program as keyed.

3	LENGTH	This field allows you to specify the length of the OS PARM field. If you do not specify a value for LENGTH, the TTF calculates the length of the OS PARM based on the position of the last non-blank character in the OS PARM field.

Debug Application Screen

Functions: The Debug Application screen is the focal point in the TTF for program execution and debugging. This screen allows you to:

- Display program areas and storage
- Set and display test program breakpoints
- Update core areas before and during program execution
- Intercept potential test program errors such as ABENDs

See "Function Area" later in this chapter, and "Commands" appendix later in this guide, for information about other functions of the Debug Application screen.

The Debug Application screen, shown below, provides information about debugging CA Telon applications.

```
DEBUG APPLICATION ****************** ****************************************
 PROGRAM  ENTRY POINT      DATE         TIME     LENGTH   PCB LIST  TRACE TABLE
TRTMMENU   00123F18    JUN 20, 2001   11.06.10   0050E8   0008649C   001204B8

                ID      ADDR           ID      ADDR          ID      ADDR
 TRACE     01 XFERWORK 00129310   05  PGM WORK 001245A0
 TABLE:    02 WRK STRG 00123FB8   06  SSA AREA 00124368
   *       03 SYS WORK 001245C0   07  TRGEMPL  0012482C
   *       04 APP WORK 00124478
 ---------------------------------   ---------------  ----------------------
    FUNCTION                          SECTION   FIELD      MODULE         DISP
  PROGRAM INIT                        Q-100           TRTMMENU.TRTMMENU 000000
00123F18   * 90ECD00C  185D05F0    4580F010  E3D9E3D4 *  * .....).0..0.TRTM
00123F28   * D4C5D5E4  E5E2D9F1    0700989F  F02407FF *  * MENUVSR1....0...
00123F38   * 96021034  07FE41F0    000107FE  00127744 *  * .......0........
00123F48   * 00123F18  00123F18    00125240  00124EC0 *  * ..............+.
00123F58   * 00125740  00127704    00000000  00000000 *  * ... ............

 TRACE:    PGM INIT Y   FILE BEFORE N - AFTER N

COMMAND ==> 1                                                SCROLL ==> 2
PAGE
          ------------------------------------------------
  AREA:     TRACE ID  3          LINE  4        DISPLAY LTH  5
   *        STOP AT ID  6                       UPDATE C/X  7
```

As shown in the example above, the screen is divided into the following areas:

- Program Information
- Trace Table
- Function Area
- Storage Display
- Trace Options
- Command Area

Each of these areas are discussed in detail below.

Program Information Area

The Program Information area consists of rows 2 and 3 on the screens. This area provides information related to the program that you are testing, including:

- **PROGRAM** - Name of the test program.
- **ENTRY POINT** - Entry point address of the test program in storage.
- **DATE** and **TIME** - Date and Time of the last compile of the test program.
- **LENGTH** - Length of the load module, in hexadecimal.
- **PCB LIST** - Address of the DL/I PCB list, if a PSB has been scheduled.
- **TRACE TABLE** - Address of the Trace Table. The TTF uses this address to display the Trace Table. This address is useful if there are more than 12 entries (more than one full screen) in the trace table. It also is useful to determine the address of the output and input buffers, which are carried in the top of the trace table.

Trace Table

The Trace Table follows the program information fields at the top of the screen, consisting of rows 5 through 9. The table shows the Program Init call for a CA Telon TSO program generated with TRACE=**Y**.

The TRACE TABLE entries in this screen include an entry number (1-7), an ID that identifies the entry uniquely, and an address of the entry in storage.

You can add entries to the Trace Table, using the TRACE ID, and LINE items in the Command Area, described later in this chapter. You can add a maximum of 32 items in the trace table, the first 12 of which are displayed on the Debug Application screen.

Function Area

The Function Area consists of rows 11 and 12 on the screens. This area displays a description of the function that caused the TTF to display the Debug Application screen as well as information related to the function.

The first field in this area, FUNCTION, identifies the reason that the Debug Application screen was invoked. The function shown is PROGRAM INIT, which results from the PROGRAM INIT trace indicator being set to **Y** to stop the test program prior to execution.

FUNCTION can display the following messages during test program execution:

PROGRAM INIT

A new test program has been loaded but has not yet begun execution. You get the Debug Application screen with the program Init function only if you specified the PROGRAM INIT indicator as **Y** on the CA Telon Test Facility Main Menu screen.

DL/I TRACE (BEFORE/AFTER)

You entered the TEST AID key or the END command key on the Trace DL/I Access screen. The TTF also displays the PARM LIST showing the address of the parameter list passed in the DL/I, I/O call.

DB2 TRACE (BEFORE/AFTER)

You entered the TEST AID key or the END command key on the Trace DB2 Access screen. The TTF also displays the PARM LIST showing the address of the parameter list passed in the DB2 access.

PFKEY

You entered the TEST AID key while the test application screen was being displayed.

STOP AT

A STOP AT break-point has been reached at the address indicated. This type of function also lists the STOP ID that established the breakpoint, as defined on the Debug Application screen.

ABEND

An abnormal termination occurred. The screen classifies the ABEND as SYSTEM or USER, and lists the following:

- COMPL CODE - The completion code associated with the ABEND
- PSW - The Program Status Word set to the address of the abending instruction

ONCODE

The TTF intercepted a PL/I ONCODE. The screen displays the oncode value and gives the retry address and ONSOURCE address, if available.

DL/I PARM INVALID

An invalid parameter has been detected in a DL/I access call. Either the function code passed is invalid, the PCB passed is invalid, or one of the other parameters passed exists in protected core. This function is displayed as nTH DL/I PARM INVALID where **n** is the number of the parameter which was detected in error.

SECTION

If you generated the test program with TRACE=**Y**, the TTF displays the program section (for COBOL) or procedure (for PL/I) that is currently processing. The TTF leaves SECTION blank if you generated the test program with TRACE=**N**.

FIELD

If you generated the test program with TRACE=**Y**, in sections/procedures where editing occurs, the TTF displays the field being edited. The TTF leaves FIELD blank if you generated the test program with TRACE=**N**. This field is useful when debugging OC7s and other field-related ABENDs.

MODULE

This identifies either the name of the program and CSECT that the TTF is displaying in the storage display area, or working storage, as is appropriate.

DISP

This is the Offset in hexadecimal, from the starting point of the MODULE.CSECT or working storage to the area that the TTF is displaying currently in the storage display area.

Storage Display Area

The Storage Display area consists of rows 13 through 17 and includes:

- Address of core, starting address in core of the 16-byte display line.
- An 80-byte area displayed in hex. When the Debug Application screen is first displayed, this area starts at the ENTRY POINT address shown at the top of the screen. You can request display of a different area of storage using the COMMAND field, as described in Appendix C, "Commands." You can modify this field, and, with the UPDATE C/X field, you can use it to alter storage. See "Command Area," later in this chapter, for more information about the UPDATE C/X field.
- An 80-byte area displayed in character. A period (.) replaces non-displayable and lowercase characters. The information in this area always corresponds to that in the Hex display area, described above. You can modify the area, and, with the UPDATE C/X field (described later in this chapter under "Command Area"), you can use it to alter storage.

Note: The TTF will not update a period (.) in the character section of the display.

Trace Options Area

The Trace Options area includes the current specifications for several of the trace options, described previously on the CA Telon Test Facility Main Menu screen. You can modify these specifications to change the option currently in effect. Use the Debug Application screen to perform these modifications. The Trace Options area consists of the following fields in row 19 on the screens:

- PGM INIT - Current specification for the PROGRAM INIT option. Valid values are **Y** or **N**.
- FILE - Current specifications for the before and after FILE ACCESS tracing option. Valid values are **Y** or **N**.

Command Area

The Command Area, consisting of rows 21 through 23 on the screens, provides various processing options, discussed below. The highlighted numbers in the COMMAND AREA correspond with the numbers below.

1	COMMAND	This field allows you to enter any of the valid test commands activated for the current test screen. Note that you can also use an address expression in this field to define an address to be displayed in the Storage Display area on the screen. See "Commands" appendix For a complete list of commands and more

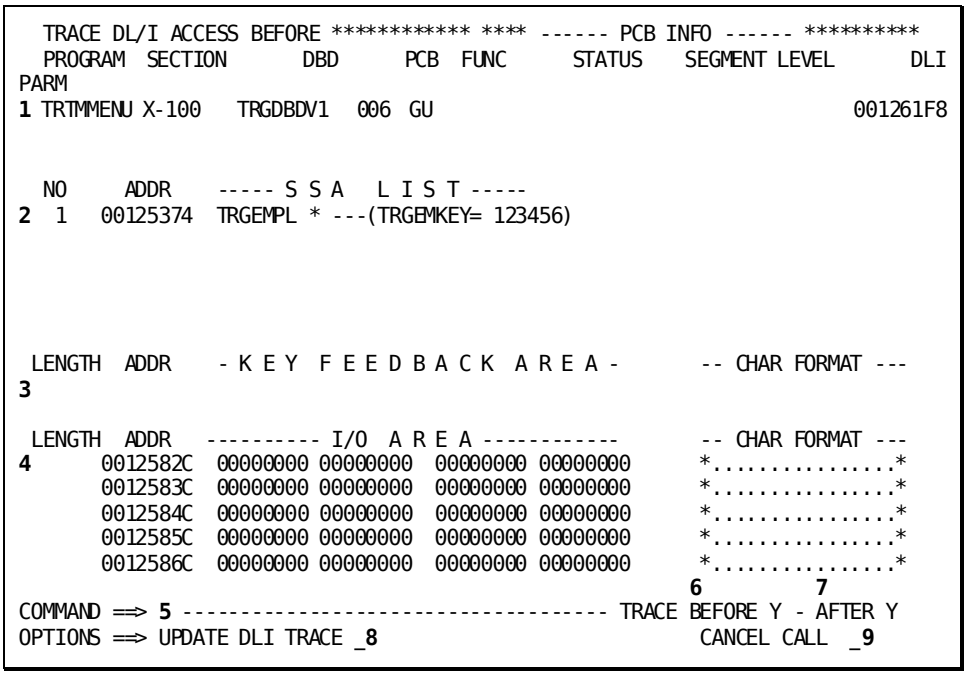
		information.
2	SCROLL	This field defines the default scroll value for use in paging commands. Options are HALF, PAGE, and a number representing the number of bytes to scroll. HALF and PAGE scroll amounts are based on the number of bytes displayed on the screen in the storage display area. Note that you can override the default SCROLL value by entering a scroll amount into the COMMAND field.
3	TRACE ID	The Trace Table contains addresses of selected areas of program storage. You can add or replace an entry in the Trace Table by specifying a name for the table entry in the TRACE ID field and its address in the COMMAND field. If you do not specify an address, the Test Facility uses the current display address. Optionally, you can specify the line at which the entry should appear in the Trace Table by entering a value in the LINE field, described below. If you do not include the line specification, the TTF adds the entry to the end of the table.
4	LINE	This field allows you to specify the line at which a requested Trace Table entry should be added to the table.
5	DISPLAY LTH	This field allows you to set the length of the data appearing in the Storage Display Area. The TTF displays 80 bytes of information by default. You can request fewer bytes by using the DISPLAY LTH field. Note that the number of bytes displayed is also used in calculating the PAGE and HALF scroll values. Once the DISPLAY LTH is set, it holds the value assigned until it is reset.

6	STOP AT ID	This field allows you to define break-points for testing purposes. Break-points are points at which program execution stops and the TTF returns the Debug Application screen. To define a break-point, enter a descriptive name in this field and enter the address at which to stop in the COMMAND field. If you do not enter an address expression in the COMMAND field, CA Telon assumes you want to stop at the address displayed currently in the display area. See the description of the STOP AT ID field, earlier in this section, for more information. All program break-points remain in effect until the Test Facility loads a new test program. However, you can reload the current program without losing the break-points, by using the RELOAD command. The AT command discussed in Appendix C offers an alternative method for setting break-points.
7	UPDATE C/X	This field allows you to modify test program storage to correspond to changes made in the hexadecimal or character display sections of the storage display area. To modify data using the hexadecimal area, enter **X**; to modify data using the character area, enter **C**.

Trace DL/I Access Screen

You access the Trace DL/I Access screen by entering **Y** for the FILE ACCESS parameter on the Test Facility Main Menu. The Trace DL/I Access screen allows you to view the arguments of a DL/I access before it is issued and allows you to check the results of the access upon completion of the call.

Optionally, you can cancel the access and return control to the test program for continued processing. The Trace DL/I Access screen is shown below as the Trace DL/I Access Before screen and the Trace DL/I Access After screen. The I/O illustrated below represents a successful GU call.



```
  TRACE DL/I ACCESS AFTER  ********** **** ------ PCB INFO ------ **********
  PROGRAM  SECTION      DBD      PCB  FUNC      STATUS    SEGMENT LEVEL      DLI
PARM
1 TRTMMENU X-100    TRGDBDV1   006  GU                         TRGEMPL  01
001261F8
  NO     ADDR    ----- S S A   L I S T -----
 2 1   00125374  TRGEMPL * ---(TRGEMKEY= 123456)




 LENGTH  ADDR    - K E Y  F E E D B A C K  A R E A -       -- CHAR FORMAT ---
 3  6  00086204  F1F2F3F4 F5F6                            *1      23456        *

  LENGTH  ADDR    ---------- I/O  A R E A ------------      -- CHAR FORMAT ---
 4 600  0012582C  F1F2F3F4 F5F6C2C9  D9C46B40 D3C1D9D9    *123456BIRD.  LARR*
        0012583C  E8404040 40404040  40404040 404040F5     *Y                5*
        0012584C  F8F1F1F2 F4D4F4F1  F3C6D3F9 F7F7F6F6      *81124M413FL9 7766*
        0012585C  D6D5C540 C3C5D3E3  C9C340D3 C1D5C540      *ONE CELTIC LANE  *
        0012586C  40404040 40404040  40C6D9C5 D5C3C840      *           FRENCH *
                                                                        6
7
 COMMAND ==> 5----------------------------------- TRACE BEFORE Y - AFTER Y
 OPTIONS ==> UPDATE DLI TRACE _8
```

For compatibility with pre-2.0 releases of the CA Telon Test Facility, Enter on the Trace DL/I Access screen acts as a GO key. This is only true if the TEST AID key is equal to one of the TEST GO keys as defined on the CA Telon Test Facility Main Menu.

Fields

The following describes the fields and their use on the Trace DL/I Access screens.

1	Program/PCB Information Fields	These fields display the following: ■ Test program information (Name and Section) ■ Relative PCB number in the PSB used in the call ■ DL/I access function code ■ PCB information for the PCB being used in the DL/I access ■ Address of the DL/I parameter list
2	SSA Fields	These fields display the count of Segment Search Arguments followed by as many as six SSAs which are being used in the DL/I call.

3	KEY FEEDBACK AREA	This area consists of fields in lines 12 through 14 on this screen. These fields show a hex and character display of as many as 32 bytes of the Key Feedback information in the PCB used in the DL/I access.
4	I/O AREA	This area consists of fields in lines 15 through 20 on this screen. These fields show the length of the segment used in the DL/I access followed by as many as 80 bytes of the segment I/O area in hex and character format.
5	COMMAND	This field allows you to enter any of the valid test commands activated for the current test screen. See "Commands" appendix for a complete list of commands and more information.
6	TRACE BEFORE	This field shows the current setting for the general File Access Trace Before indicator. You can update this field with either **Y** (Trace Access Before) or **N** (do not Trace Access Before).
7	TRACE AFTER	This field shows the current setting for the general File Access Trace After indicator. You can update this field with either **Y** (Trace Access After) or **N** (do not Trace Access After).
8	UPDATE DL/I TRACE	This field allows you to transfer to the Update DL/I Trace screen where you can specify trace requirements on the PSB or PCB level. Enter any non-blank character to transfer to the Update DL/I Trace screen.
9	CANCEL CALL	This field appears only when the TTF displays Trace DL/I Access Before screen. Enter any non-blank character in this field to continue program execution without issuing the access to DL/I.

Trace DB2 Access Screen

You access the Trace DB2 Access screen by entering **Y** for the FILE ACCESS parameter on the Test Facility Main Menu. The Trace DB2 Access screen allows you to view the arguments of a DB2 access before it is issued and allows you to check the results of the access upon completion of the call. You can use the Trace DB2 Access Before screen to cancel the access and return control to the test program for continued processing.

The Trace DB2 Access Before screen and the Trace DB2 Access After screen are shown below. The I/O illustrated represents a successful INSERT command.

```
 TRACE DB2 ACCESS BEFORE ************ ************************* PAGE 01 *****
1 PROGRAM  SECTION          COMMAND   SQLCODE   COLCNT               DB2 PARM
  TRTMCOMB U-100-11         INSERT               012                 00036864

2 COL LTH   FROM   ---------  I/O  A R E A S ---------  -- CHAR FORMAT --- TYPE
  001 006 00035E48 F1F2F3F4 F5F6                       *123456          * CHR
  002 030 00035E4E C2C9D9C4 6B40D3C1  D9D9E840         *BIRD. LARRY     * CHR
  003 004 00035E6C 0550921C                            *.&..            * DEC
  004 001 00035E70 D4                                  *M               * CHR
  005 010 00035E71 F3F0F7F4 F2F1F8F7  F7F2             *3074218772      * CHR
  006 030 00035E7B D6D5C540 C3C5D3E3  C9C340D3  C1D5C540 *ONE CELTIC LANE * CHR
  007 030 00035E99 C6D9C5D5 C3C840D3  C9C3D240  40404040 *FRENCH LICK     * CHR
  008 002 00035EB7 C9D5                                *IN              * CHR
  009 005 00035EB9 F9F7F7F6 F5                         *97765           * CHR
  010 004 00035EBE 0820817C                            *....            * DEC
  011 003 00035EC2 F0F0F1                              *001             * CHR
  012 003 00035EC5 10000C                              *...             * DEC
  013 003 00035EC8 00350C                              *...             * DEC


                                                                  4      5

 COMMAND ==> 3------------------------------------ TRACE BEFORE Y - AFTER Y
 OPTIONS ==> UPDATE DB2 TRACE _6                    CANCEL CALL _7

 TRACE DB2 ACCESS AFTER  ************ ************************* PAGE 01 ****
1 PROGRAM  SECTION          COMMAND   SQLCODE   COLCNT                DB2 PARM
  TRTMCOMB U-100-11         INSERT     +000      013                  00036864

2 COL LTH   FROM   ---------  I/O  A R E A S ----------  -- CHAR FORMAT--- TYPE
  001 006 00035E48 F1F2F3F4 F5F6                       *123456          * CHR
  002 030 00035E4E C2C9D9C4 6B40D3C1  D9D9E840 40404040  *BIRD. LARRY     * CHR
  003 004 00035E6C 0550921C                            *.&..            * DEC
  004 001 00035E70 D4                                  *M               * CHR
  005 010 00035E71 F3F0F7F4 F2F1F8F7  F7F2             *3074218772      * CHR
  006 030 00035E7B D6D5C540 C3C5D3E3  C9C340D3  C1D5C540 *ONE CELTIC LANE * CHR
  007 030 00035E99 C6D9C5D5 C3C840D3  C9C3D240  40404040 *FRENCH LICK     * CHR
  008 002 00035EB7 C9D5                                *IN              * CHR
  009 005 00035EB9 F9F7F7F6 F5                         *97765           * CHR
  010 004 00035EBE 0820817C                            *....            * DEC
  011 003 00035EC2 F0F0F1                              *001             * CHR
  012 003 00035EC5 10000C                              *...             * DEC
  013 003 00035EC8 00350C                              *...             * DEC


                                                                  4      5

  COMMAND ==> 3------------------------------------ TRACE BEFORE Y - AFTER Y
OPTIONS ==> UPDATE DB2 TRACE _6
```

For compatibility with pre-2.0 releases of the CA Telon Test Facility, the Enter on the Trace DB2 Access screen acts as a GO key. This is only true if the TEST AID key is equal to one of the TEST GO keys as defined on the CA Telon Test Facility Main Menu. You access the Trace DB2 Access screen by entering **Y** for the FILE ACCESS parameter on the Test Facility Main Menu.

Fields

The following describes the fields and their use on the Trace DB2 Access screens.

1	Program/DB2 Information Fields	These fields display the following: ■ Test program information (Name and Section) ■ DB2 command ■ Resultant SQLCODE from the access ■ Number of DB2 columns for display on the screen ■ Address of the parameter (PLIST) passed to DB2
2	Column Fields List	These fields are the columns used in the DB2 access. The columns relative number is displayed followed by its lth, address, hex value, character value, and type. Note that you can use the scroll keys if more than 15 columns are available for display. The screen displays a page number and MORE in the upper right to indicate the current page and if additional pages exist.
3	COMMAND	This field allows you to enter any of the valid test commands activated for the current test screen. See "Commands" appendix for more information and a complete list of commands.
4	TRACE BEFORE	This field shows the current setting for the general File Access Trace Before indicator. You can update this field with either **Y** (Trace Access Before) or **N** (do not Trace Access Before).
5	TRACE AFTER	This field shows the current setting for the general File Access Trace After indicator. You can update this field with either **Y** (Trace Access After) or **N** (do not Trace Access After).
6	UPDATE DB2 TRACE	This field allows you to transfer to the Update DB2 Trace screen. Enter any non-blank character to transfer to the screen.
7	CANCEL CALL	This field appears only when the TTF displays the Trace DB2 Access Before screen. Enter any non-blank character in this field to continue program execution without issuing the access to DB2.

Update DL/I Trace Screen

The Update DL/I Trace screen displays the scheduled DL/I PSB and its PCBs. This screen allows trace selection on the PSB and PCB level. Trace selection occurs in the following sequence:

1. Trace selection at the PCB level takes precedence over trace selection at the PSB level
2. Trace selection at the PCB level takes precedence over the general purpose FILE ACCESS TRACE indicators displayed on the CA Telon Test Facility Main Menu, Debug Application, and through Trace Access screens

You access the Update DL/I Trace screen in the following ways:

- Select the UPDATE PARAMETER LIST option for the DL/I parameter list format on the CA Telon Test Facility Main Menu
- Select the UPDATE DL/I TRACE option on the Trace DL/I Access screen
- Issue the Trace DL/I command, when valid, from any of the other TTF screens

To leave this screen, use the TEST AID key or enter either one of the following commands:

- **GO**
- **END**
- **MENU**
- **=X**

```
UPDATE DL/I TRACE ***************** ****************************************
 COMMAND ==> 1
             ---------------------------------------------------- PAGE 01

                                                 TOTAL
 PSB      PSBNAME BEF AFT FUNCTION COUNT CALLS
  *       TRPSBCT  2   3  4_______ 5___ _ 00006    6

 PCB      LTERM/                             TOTAL
 NBR TYPE DBDNAME BEF AFT FUNCTION COUNT CALLS  ST   SEG    LVL KEY FEEDBACK
 001  IO  SMITH    7   8  9        10    00000  11
 002  TP           _   _  ________ _____ 00000
 003  TP  MPRINT   _   _  ________ _____ 00000
 004  DB  TRGDBDV1 _   _  ________ _____ 00006  GE TRGEMPL  01  123456
 005  DB  HLPDBDV1 _   _  ________ _____ 00000
 006  DB  HLDDBDV1 _   _  ________ _____ 00000
 007  DB  WKSDBDV1 _   _  ________ _____ 00000
```

Fields

The following describes the function and usage of each field. The numbers in fields correspond to the numbers below.

1	COMMAND	This field allows you to enter any of the valid test commands activated for the current test screen. See "Commands" appendix for a complete list of commands and more information.
2	PSB BEFORE	This field allows you to request BEFORE ACCESS tracing of access to all of the PCB's in the PSB. Valid values are: ■ **Y** - Request BEFORE ACCESS tracing ■ **N** - Turn off BEFORE ACCESS tracing A value entered in this field overrides any values that you specify on the general FILE ACCESS TRACE indicators previously mentioned. If you leave this field blank, the TTF uses the general FILE ACCESS BEFORE indicator. You can override this specification at the PCB level.
3	PSB AFTER	This field allows you to request AFTER ACCESS tracing of access to all of the PCBs in the PSB. Valid values are: ■ **Y** - Request AFTER ACCESS tracing ■ **N** - Turn off AFTER ACCESS tracing A value entered in this field overrides any values that you specify on the general FILE ACCESS TRACE indicators. If you leave this field blank, the TTF uses the general FILE ACCESS AFTER indicator. You can override this specification at the PCB level.
4	PSB FUNCTION	This field allows you to limit further the trace of before- and after-accesses to specific function codes. You can specify DL/I function codes or partial function codes suffixed by an asterisk. For example, **'GH*,ISRT'** requests tracing of all DL/I GET HOLD and INSERT calls. You must separate function codes in this field by either a comma or one or more spaces. You can override this specification at the PCB level. **Note:** The selected functions are used to further limit the tracing defined by BEFORE and AFTER trace indicators on any level, and that you must set one or both of these values to **Y** for tracing to occur.

5	PSB COUNT	This field allows you to specify the number of calls at which the trace BEFORE and AFTER indicators at the PSB level are to be set automatically to **Y**. When the PSB TOTAL CALLS value reaches this number, the indicators are set to **Y** regardless of their current value. DL/I access tracing then begins for all PCBs.
6	PSB TOTAL CALLS	This field shows the total accesses against the PSB, that is, the sum of all the calls against all of the PCBs since execution of the current test program began in the TSO environment. This field is for informational purposes and you can use it during debugging to set the PSB COUNT field in subsequent testing executions.
7	PCB BEFORE	This field allows you to request BEFORE ACCESS tracing of access to this PCB Valid values are: ■ **Y** - Request BEFORE ACCESS tracing ■ **N** - Turn off BEFORE ACCESS tracing The value specified in this field overrides any values specified in either the PSB ACCESS BEFORE TRACE or the general FILE ACCESS BEFORE TRACE fields. If you leave this field blank, the TTF uses the next higher-level indicator.
8	PCB AFTER	This field allows you to request AFTER ACCESS tracing of access to this PCB Valid values are: ■ **Y** - Request AFTER ACCESS tracing ■ **N** - Turn off AFTER ACCESS tracing The value specified in this field overrides any values specified in either the PSB ACCESS AFTER TRACE or the general FILE ACCESS AFTER TRACE. If you leave this field blank, the TTF uses the next higher-level indicator.
9	PCB FUNCTION	This field allows you to limit further the trace of before- and after-accesses to specific function codes. You can specify DL/I function codes or partial function codes suffixed by an asterisk. For example, **'GH*,ISRT'** requests tracing of all DL/I GET HOLD and INSERT calls. You must separate function codes in this field by either a comma or one or more spaces. **Note:** The selected functions are used to limit further the tracing defined by BEFORE and AFTER trace indicators on any level, and that you must set one or both of these values to **Y** for tracing to occur.

10	PCB COUNT	This field allows you to specify the number of calls at which the trace BEFORE and AFTER indicators at the PCB level are to be set automatically to **Y**. When the PSB TOTAL CALLS value reaches this number, the indicators are set to **Y** regardless of their current value. DL/I access tracing then begins for all PCBs.
11	PCB TOTAL CALLS	This output field shows the total accesses against the PCB since execution of the current test program began in the TSO environment. This field is for informational purposes and you can use it during debugging to set the PCB COUNT field in subsequent testing executions.

Update DB2 Trace Screen

The Update DB2 Trace screen allows you to:

- Specify before/after tracing of DB2 calls
- Select specific SQL commands or partial commands for tracing
- Specify a count value after which to show the Trace DB2 Access screen

You access the Update DB2 Trace screen in the following ways:

- Select the UPDATE DB2 TRACE option on the Trace DB2 Access screen
- Issue the Trace DB2 command, when valid, from any other TTF screen

To exit this screen, use the TEST AID key or enter either the **GO**, **END**, **MENU**, or **=X** commands.

```
UPDATE DB2 TRACE ****************** ****************************************
COMMAND ==> 1
            ------------------------------------------------------------------



                                                                      TOTAL
DB2      BEF AFT             DB2 COMMANDS                    COUNT    CALLS
ACCESS    2   3  4_____________________________________  5___ 00017  6
```

Fields

The following describes the function and usage of the fields. The numbers in the fields correspond to the numbers below.

1	COMMAND	This field allows you to enter any of the valid test commands activated for the current test screen. See "Commands" appendix for a complete list of commands and more information.
2	DB2 BEFORE	This field allows you to request BEFORE ACCESS tracing of access to all DB2 tables. Valid values are: ■ **Y** - Request BEFORE ACCESS tracing ■ **N** - Turn off BEFORE ACCESS tracing The value specified in this field overrides any values specified on the general FILE ACCESS TRACE indicators previously mentioned. If you leave this field blank, the TTF uses the general FILE ACCESS BEFORE indicator.
3	DB2 AFTER	This field allows you to request AFTER ACCESS tracing of access to all DB2 tables. Valid values are: ■ **Y** - Request AFTER ACCESS tracing ■ **N** - Turn off AFTER ACCESS tracing The value specified in this field overrides any values specified on the general FILE ACCESS TRACE indicators previously mentioned. If you leave this field blank, the TTF uses the general FILE ACCESS AFTER indicator.
4	DB2 COMMANDS	This field allows you to limit further the trace of before- and after- accesses to specific commands. You can specify DB2 SQL commands or partial commands suffixed by an asterisk. For example, **'S*,INSERT'** requests tracing of all DB2 SELECT and INSERT calls. You must separate commands in this field by either a comma or one or more spaces. **Note:** The selected commands are used to further limit the tracing defined by BEFORE and AFTER trace indicators on any level, and that you must set one or both of these values to **Y** for tracing to occur.

5	DB2 COUNT	This field allows you to specify the number at which the trace BEFORE and AFTER indicators for DB2 are to be automatically set to **Y**. When the DB2 TOTAL CALLS value reaches the number specified in this field, the indicators are set to **Y** regardless of their current value. DB2 access tracing then begins.
6	DB2 TOTAL CALLS	This field displays the total accesses to DB2 tables since execution of the test program began.

Display Full Screen Storage Screen

The Display Full Screen Storage screen is an extension of the core area displayed on the Debug Application screen previously mentioned. The core area in this screen displays as many as 04 bytes of storage and displays this amount by default unless you have reset the DISPLAY LTH field value on either this screen or the Debug Application screen. You access the Display Full Screen Storage screen by issuing the DISPLAY STORAGE (D S) command from any of the other TTF screens on which the command is valid.

To exit this screen, use the TEST AID key or enter either the **GO**, **END**, **MENU**, or **=X** commands.

```
DISPLAY FULL SCREEN STORAGE ********* ****************************************
COMMAND ==>  1_____________________________________   SCROLL 2 ==>PAGE
                                                    MODULE       DISP 1
  AREA:      UPDATE C/X 3   DISPLAY LTH 4              5  6 000000
                                                  TRTMMENU.TRTMMENU
00123F18   * 90ECD00C  185D05F0    4580F010  E3D9E3D4 *  * .....).0..0.TRTM
00123F28   * D4C5D5E4  E5E2D9F1    0700989F  F02407FF *  * MENUVSR1....0...
00123F38   * 96021034  07FE41F0    000107FE  00127744 *  * .......0........
00123F48   * 00123F18  00123F18    00125240  00124EC0 *  * ..............+.
00123F58   * 00125740  00127704    00000000  00000000 *  * ... ............
00123F68   * 00000000  00000000    00000000  00000000 *  * ................
00123F78   * 00000000  00000000    00000000  00000000 *  * ................
00123F88   * 00000000  00000000    00000000  00000000 *  * ................
00123F98   * 00000000  001250F0    F1F14BF0  F64BF1F0 *  * ......&011.06.10
00123FA8   * D1E4D540  F2F06B40    F1F9F8F4  00000000 *  * JUN 20. 2001....
00123FB8   * E3C5D3D6  D540C9C4    C5D5E340  40F14BF2 *  * TELON IDENT  1.2
00123FC8   * F8F3F1F1  F1F5F0F0    F3F7F8F4  F0F5F1F5 *  * 8311150037840515
00123FD8   * C3C9C940  40400000    00000000  00000000 *  * CII   .........
00123FE8   * E340C540  D340D640    D5404040  E340D940 *  *  T E L O N  T R
00123FF8   * C140C940  D540C940    D540C740  4040E240 *  *  A I N I N G  S
00124008   * E840E240  E340C540    D4D7D9C9  D4C1D9E8 *  *  Y S T E MPRIMARY
00124018   * 40D4C5D5  E4E2C5D3    C5C3E340  D6D7E3C9 *  *   MENUSELECT OPTI
00124028   * D6D5C5D4  D7D3D6E8    C5C540C9  C4C5D4D7 *  *  ONEMPLOYEE IDEMP
00124038   * D3D6E8C5  C5404040    F14B40C1  C4C4F24B *  *  LOYEE   1. ADD2.
```

Fields

The following describes the function and usage of each field. The numbers in the fields correspond to the numbers below.

1	COMMAND	This field allows you to enter any of the valid test commands activated for the current test screen. Note that you can also use an address expression in this field to define an address to be displayed in the storage display area on the screen. See "Commands" appendix for a complete list of commands and more information.
2	SCROLL	This field defines the default scroll value for use in paging commands. Options are: ■ HALF ■ PAGE ■ The number of bytes to scroll HALF and PAGE scroll amounts are based on the number of bytes displayed on the screen in the storage display area. You can override the default value in this field by entering a scroll amount in the COMMAND field.
3	UPDATE C/X	This field allows you to modify test program storage to correspond to changes made in the hexadecimal or character display section of the storage display area. Valid values are: ■ **X** - Modify data using the hexadecimal area ■ **C** Modify data using the character area
4	DISPLAY LTH	This field allows you to set the length of the data appearing in the Storage Display Area. The default display length for the TTF is 302 bytes of information. You can use this field to request fewer bytes. The TTF also uses this number when calculating the PAGE and HALF scroll values. This field retains the specified value until you enter a new value.
5	MODULE	This field identifies either the name of the program and CSECT that the TTF displays in the storage display area, or working storage, as appropriate.
6	DISP	This field displays the offset in hexadecimal, from the starting point of the MODULE CSECT or working storage to the area that the TTF displays in the storage display area.

List Module Map Screen

The List Module Map screen displays the link-edit map for the test program being tested. This screen also displays link-edit maps for any module loaded in the TTF address space. Scrolling of the Module Map is available.

You access the List Module Map screen by:

- Issuing the LISTMAP command from any other TTF screen
- Selecting the MAP ENTRY NUMBER option from the List TSO Loaded Modules screen

To leave this screen, use the TEST AID key or enter one of the following commands:

- **GO**
- **END**
- **MENU**
- **=X**

As shown in the example below, a page number and a MORE indicator are located in the upper right corner of the screen to display the current page and scrolling availability.

```
LIST MODULE MAP ******************** ****************************************
COMMAND ==>  1                                                  PAGE 01 MORE
             ---------------------------------------------------
OPTIONS ==>  DISPLAY LISTLOAD  2
                             ---
                        MAP OF MODULE TRTMMENU
           3          4          5              6        7       8
          CSECT       ADDR     LENGTH             ENTRY      ADDR     OFFSET
         TRTMMENU    00123F18   0038B6
         ADGADBER    001277D0   000050
         ADTAXCTL    00127820   00000A
         DFSLI000    00127830   000098           DFSPLI    00127838   000008
                                                 PLITDLI   00127838   000008
                                                 CBLTDLI   0012785C   00002C
                                                 DFSCOBOL  0012785C   00002C
                                                 DFSFOR    00127880   000050
                                                 FORTDLI   00127880   000050
                                                 DFSASM    001278A4   000074
                                                 ASMTDLI   001278A4   000074
         IFORMAT     001278C8   000138
         IFULLNUM    00127A00   000204
         ILALPHA     00127C08   0004E8
         ILBOSRV     001280F0   0004A4           ILBOSRV0  001280FA   00000A
                                                 ILBOSR5   001280FA   00000A
```

Fields

The following describes the function and usage of each field. The numbers in the fields correspond to the numbers below.

1	COMMAND	This field allows you to enter any of the valid test commands activated for the current test screen. See "Commands" appendix for a complete list of commands and more information.
2	DISPLAY LISTLOAD	This field is available only in the TSO environment. Enter any non-blank character in this field to transfer to the List TSO Loaded Modules screen.
3	CSECT	This field displays the control section (subroutine) name within the composite module displayed.
4	ADDR	This field displays the core address of the CSECT (subroutine) within the composite module displayed.
5	LENGTH	This field displays the length of the CSECT (subroutine) in hexadecimal.
6	ENTRY	This field displays an alternate entry point that is defined in the CSECT displayed.
7	ADDR	This field displays the core address of the ENTRY point within the CSECT displayed.
8	OFFSET	This field displays the offset in hexadecimal of the ENTRY point from the beginning of the CSECT displayed.

List TSO Loaded Modules Screen

The List TSO Loaded Modules screen displays a list of modules loaded in the CA Telon Test Facility's address space along with their entry points and lengths. Scrolling of the TSO Loaded Modules is available on this screen.

You access the List TSO Loaded Modules screen in any of the following ways:

- Issue the LISTLOAD command from any other TTF screen
- Select the DISPLAY LISTLOAD option from the List Module Map screen

To leave this screen, use the TEST AID key or enter one of the following commands:

- **GO**
- **END**
- **MENU**
- **=X**

As shown in the example below, the screen displays a page number and MORE in the upper right to indicate the current page and if additional pages exist.

```
LIST TSO LOADED MODULES ************ ***************************************
COMMAND ==>  1                                                  PAGE 01 MORE
             ------------------------------------------------------
OPTIONS ==>  MAP ENTRY NUMBER  2
                               ---
                      LOAD LIST OF TCB AT 007C03A8  3
          4              5       6
          MODULE         EPA     LENGTH              MODULE     EPA      LENGTH
   01   TRTMMENU    00123F18   0050E8       17   DFSDBH00  00089838   0057C8
   02   IGG019BB    00F1FAF8   000000       18   DFSBML00  0003D1F0   000250
   03   IGG019BA    00C0B1F8   000000       19   DFSA0S80  0007B530   000810
   04   IGG0193B    00CD64C8   000000       20   DFSA0S60  00080F38   0020C8
   05   TSMASDEF    0003C538   0003E8       21   DFSA0S10  000590A0   000CE0
   06   DFSFLST0    0003D7A0   000860       22   DFSPR000  0003C920   0006E0
   07   DFSDVSM0    000E75F8   006A08       23   DFSPCC30  000535E0   0008F8
   08   DFSDVBH0    000E5978   001688       24   DFSBATWK  000510B8   0005B8
   09   DFSBFSPP    000E0000   005000       25   DFSCNS00  00051670   000590
   10   DFSWEQEL    00080738   008000       26   DFSCNSWK  00051C00   000320
   11   DFSDXMT0    00083178   001E88       27   DFSCSS00  0007A048   000FB8
   12   DFSDLR00    000A8280   009D80       28   DFSCSSWK  00049248   000320
   13   DFSDL0C0    000A32C8   004D38       29   DFSCST00  0004E028   000570
   14   DFSDLD00    0009ABC0   008440       30   DFSG0001  00094700   001000
   15   DFSDHDS0    00095BE8   004418       31   DFS0S001  00079000   001000
   16   DFSDDLE0    0008FD28   0052D8       32   DFSGI001  00948000   001000
```

Fields

The list below provides descriptions of the function and usage of each field. The number next to each field on the sample screen corresponds to the number of each field listed below.

1	COMMAND	This field allows you to enter any of the valid test commands activated for the current test screen. See "Commands" appendix for a complete list of commands and more information.
2	MAP ENTRY NUMBER	This field allows you to transfer to the List Module Map screen and then display the link-edit map for the selected module if the map is available. Enter the module number (numbers 1 through 32) for the desired link-edit map display.
3	TCB Address Field	This field provides the address of the Task Control Block from which the Load List was generated.
4	MODULE	This field displays the name of the composite module loaded into the TTF address space.
5	EPA	This field displays the address of the displayed module's entry point.
6	LENGTH	This field displays the length in hexadecimal of the displayed composite module.

Chapter 3: Advanced Procedures

This chapter contains step-by-step instructions to perform several useful functions with the CA Telon Test Facility. The procedures are intended to guide you through the steps needed.

This chapter illustrates how to:

- Set break points
- Interrogate storage
- Locate PL/I static variables
- Add a value to a trace table
- Test field edits

See the "Sample Session" chapter for an example of how these separate procedures fit together and for more information.

Setting Break Points

The TTF allows you to interrupt execution of the application program and inspect and change storage. You can automatically stop the program at the program load point, before and after data access, and when the application screen displays on the terminal. Setting other stop points is useful for debugging ABENDs and logical errors that occur elsewhere in the program. You can set stop points on any instruction in the program.

To set a stop or break point, perform the following steps:

1. Compile the program you want to test with a parameter that provides the offset of each instruction. For COBOL, use PMAP or CLIST for PL/1, use LIST.
2. Determine the offset of the instruction where you want to set the stop.
3. Run the TTF and stop the program at load (PROGRAM INIT=**Y**).
4. Enter the ENTRY POINT address (displays on the upper-left of the debug screen) as an additional line in the trace table. Add it as follows:
   - ADDR/LINE: leave blank
   - TRACE ID: LOAD
   - LINE: next free trace table line number

5. Determine the stop points as follows:

   - For COBOL, find the statement offset in your COBOL PMAP (this is the offset from load point). On the TTF COMMAND field, key in the above entry point plus the offset of the stop instruction. For example, type **166D30 + 4B6** or set ADDR/LINE: line number of LOAD + offset of statement and press **ENTER**.
   - For PL/I, the object code is in a CSECT. Name the CSECT by adding a suffix of **1** to the main procedure. Enter **load module.CSECT name +offset** to set the stop address. For example, for HEADER TR and screen ID. JU01, the program name is TRxJU01 and the CSECT is TRxJU011. If the instruction to stop at has an offset of 4B6, Enter **TRxJU01.TRxJU011.+4B6** in the TDF command field. This positions the program on that statement.

6. Enter a symbolic name (any name) into the STOP AT ID field and press **ENTER** to establish the stop point. The program stops at this point during execution and displays the STOP AT ID **symbolic name**.

   Note: The program is interrupted **before** the relevant statement executes.

7. Write the address of the stop point since it only appears in the trace table while that program is loaded. Once another program is accessed, it is lost from the trace table. However, if the program is reloaded, the address remains unchanged and can be re-entered into the trace table and the STOP point re-established.

   Note: To set more than one stop point in a program, it is easier to first create a Trace Table id for the entry point. For COBOL, type the entry point address in the TTF command field. For PL/I, Enter **load module.CSECT name.** in the command field. Enter a name such as **ENTRY** in the TRACE ID field and press **ENTER**. ENTRY is added to the Trace Table list with a Trace id line number. From then on, to set a stop you can Enter **# + offset** for the address.

   For example, if the Trace Table list contains the following:

```
          ID        ADDR
12      ENTRY      16234C
```

   Enter 12+4B6 to set the stop point from the examples above.

8. Press the TEST GO key or type **GO** on the COMMAND field after you set all stop points. Press **ENTER** to start the program execution.

Whenever the program passes through a stop point during execution, it stops before executing the statement. You can then use the debug screens, and display or change working storage before you type **GO** or press the TEST GO key to continue program execution.

If you set more than one stop point, you can detect where the program stopped from the symbolic name you gave each stop point. One way to detect where the program stopped is to make the symbolic name the program's statement number. The symbolic name appears on the middle left side of the screen, under the FUNCTION field.

Inspecting Storage

You can inspect or interrogate storage at any of the stop points you indicate in your program. Descriptions of inspecting storage for COBOL, COBOL II and PL/I follow, varying slightly between languages. The TTF works the same way but locating the address of the variable differs.

COBOL II and COBOL for z/OS

Inspect storage in COBOL by performing the following steps:

1. Compile the program with the DMAP parameter.
2. Find the BLW cell and offset of the variable you want to inspect in the Data Division Map.
3. Use the TTF to stop the program at the appropriate point. For example, at Load point find the Trace Table entry for working storage. To see a host variable in working storage, Enter its address in the TTF command field by Entering **#+BF**, where:
   - **#** is the line number of working storage in the Trace Table
   - **B** is the BLW
   - **F** is the offset of the variable from the MAP.

   You do not need to do the addition.

   For example, if the Trace Table list contains:

```
          ID          ADDR
03     WRK STRG     12F008
```

   and the variable you require (LAST-LINENO for example) is in BL=1 with offset E26, then Enter **03+0E26** (or **3+E26**). Another variable (DFHEIV4 for example) is in BL=2 and offset 3C8, Enter **3+13C8**. You do not need to Enter quotes.

PL/I

Inspect storage in PL/I by performing the following steps:

1. Compile the program with the MAP and OFFSET parameters
2. Find the offset of the variable you want to inspect
3. Use the test facility to stop the program at the appropriate point

Static storage usually starts at the address stored in register three. To see a variable in working storage, Enter **R3+F**, where **F** is the offset. The CSECT, identified by program name suffixed with a **2**, contains static storage. This name plus the variable offset also displays the variable. For example, **TRTJU012+4B6** in the Command field.

For automatic storage variables, use the TTF AUTOMAP command and the offset of the variable to determine its location in working storage.

Adding a Field to the Trace Table

Add a field to the trace table by Entering information in the following fields:

- COMMAND - Address of register + offset of field (without high-level zeros)
- TRACE ID - Any symbolic name desired
- LINE - Next free line number in the table

When you press **ENTER**, the field is added to the trace table. You can access this field in any of the usual ways (by Entering a line number or symbolic name in the COMMAND parameter) as long as this program is loaded in the test facility.

See Modify the Contents of Storage in the "Sample Session" chapter later in this guide, for information about changing the contents of storage.

Locating PL/I Static Variables

To locate static variables in a program running under the CA Telon Test Facility, perform the following steps:

- Enter **R3** on the COMMAND line when you Enter the CA Telon Test Facility Debug screen. Register 3 contains the start of static storage.
- Enter this address into the Trace Table.
- Add the hex offset of the desired variable to the address saved in the trace table. Use the variable storage map produced from the compile to do this. This gives the desired variable.

Examples

From the variable storage map, variable A has a hex offset of 748, and variable B has a hex offset of E79.

Enter the CA Telon Test Debug screen and Enter the following information:

- **R3** on the COMMAND field. The address in Register 3 is now the first display address. This is the start of static storage.
- A meaningful name (such as STATSTG.STATSTG) on TRACE ID. Add it to the Trace Table as the next sequence number (i.e., 07).
- Enter **+748** on the COMMAND field. The address of variable A is the first address displayed, and the contents of variable A in core are displayed.
- Enter the sequence number of static storage (for this example 07) on the COMMAND field. This returns you to the address of static storage.
- Enter **+E79** on the COMMAND field. The address of variable B is the first address displayed, and the contents of variable B in core are displayed.

Testing Field Edits

The Test Facility allows you to test the field edits, or subroutines, that you include in your CA Telon applications.

CA Telon programs use field edits during the editing and formatting of data fields. They are used either on output when fields are moved from a file or work area to a screen, or on input when fields are moved from the screen to a file or work area. On output, a field edit simply reformats the field as it is moved. On input, a field edit first checks the field for valid format/content before reformatting it for storage.

Field edit subroutines perform these field edits. CA Telon supplies several standard routines. You can also code your own routines in COBOL, PL/I or Assembler. See the *Design Facility Reference guide* for complete specifications for field edits. This section tests existing field edits for illustration.

Writing Your Own Field Edits

Field edits that you write must be either input field edits or output field edits. Input field edit names must begin with **I**. Output field edit names must begin with **O**. The first letter tells CA Telon how to process the edits. If the field edit name does not begin with I or O, CA Telon returns the message **EDIT NAME MUST BEGIN WITH 'I' OR 'O'**.

When you write customized field edits, they must conform to specific standards for the following items:

- Module names
- Parameters received by the module
- Process flow within a module
- Implementation guidelines

See the *Design Facility Reference guide* for details about standards for these items.

Inserting Field Edits into the Program

You insert field edits into your program through the panel definition. Enter the name of the field edit in the FLDTYPE parameter on the Update Panel Definition screen. CA Telon will then call the subroutine when it encounters the field during processing.

Getting Started

When you want to test field edits, invoke the CA Telon Test Facility. See Chapter 2, "Screen Descriptions," earlier in this manual, for complete instructions for invoking and running the TTF.

You begin testing at the CA Telon Test Facility Main Menu, as discussed below.

Input Field Edits

Input field edits check the field for valid format and/or content before reformatting it for storage. These field edits ensure that the program does not attempt to process invalid data.

In the example below, the input field edit IZIP ensures that the zip code Entered is 15 bytes long:

- Enter **X** in the TEST FIELD EDIT field.
- Enter the name of the test field edit in the TEST PROGRAM NAME field. Since this is an input field edit, the name begins with **I**.
- Press Enter.

```
TELON TEST FACILITY MAIN MENU ******* ****************************************
  COMMAND ==> ____________________________________________________________
_
  OPTIONS ==> TEST FIELD EDIT X      UPDATE PARAMETER LIST _
                                     PARAMETER LIST FORMAT DLI  (OS,DLI)

    TEST PROGRAM NAME  IZIP___

    ---------------------- TRACE OPTIONS --------------------------
                                                  BEFORE  AFTER
    STOP AT PROGRAM INIT  N        FILE ACCESS      N       N


    MISCELLANEOUS OPTIONS:                 SYSTEM INFORMATION:
    ---------------------------------      ---------------------------------
    TEST AID KEY PF12                      LTERM   TDEMO1A_    DATE  09/24/90
    TEST GO KEYS PF10 PF22                 USERID  TDEMO1A_    TIME  16:10:03
    TSO TRANSFER WORK AREA SIZE   32 K
    REFRESH WORKING STORAGE Y
```

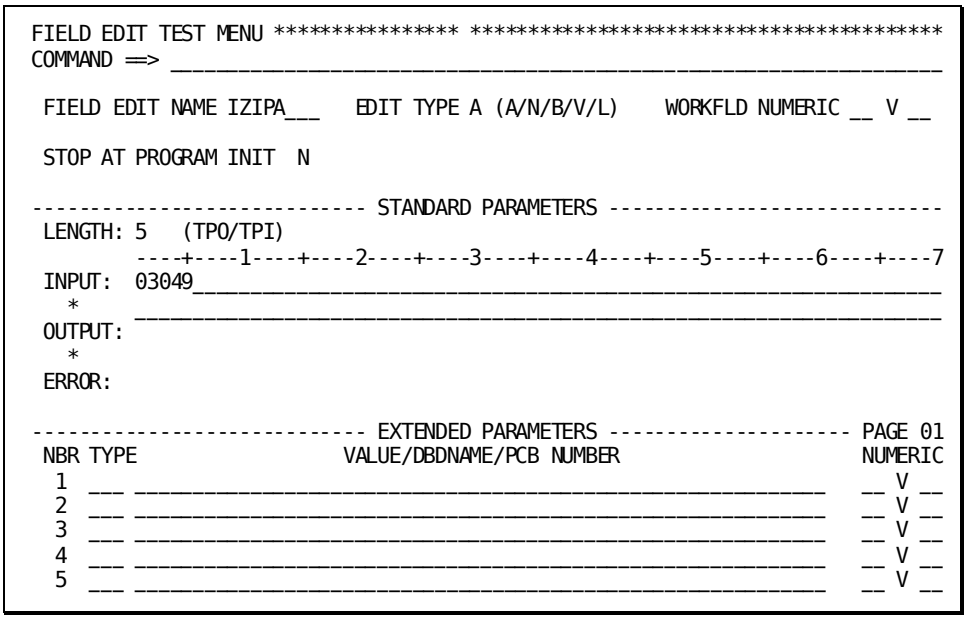
The TTF returns the Field Edit Test Menu. See the "Screen Descriptions" chapter earlier in this manual, for more information about the fields on the Field Edit Test Menu.

As shown in the example below, the name of the test field edit, IZIP, carries over to this screen.

```
FIELD EDIT TEST MENU ********** ************************************
COMMAND ==> _____________________________________________________________

 FIELD EDIT NAME IZIP____   EDIT TYPE _ (A/N/B/V/L)    WORKFLD NUMERIC __ V __

 STOP AT PROGRAM INIT  N

---------------------------- STANDARD PARAMETERS ----------------------------
 LENGTH:   0 (TPO/TPI)
         ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
 INPUT:  _____________________________________________________________________
   *     _____________________________________________________________________
 OUTPUT:
   *
 ERROR:

---------------------------- EXTENDED PARAMETERS --------------------- PAGE 01
 NBR TYPE                VALUE/DBDNAME/PCB NUMBER                     NUMERIC
  1  ___ ____________________________________________________________   __ V __
  2  ___ ____________________________________________________________   __ V __
  3  ___ ____________________________________________________________   __ V __
  4  ___ ____________________________________________________________   __ V __
  5  ___ ____________________________________________________________   __ V __
```

You must specify the following additional information about the edit for testing on the Field Edit Test Menu:

- Enter **A** for alpha in the EDIT TYPE field. Other valid edit types are:
  - **N** - Numeric
  - **B** - Binary
  - **U** - Unable
  - **L** - Lowercase
- Enter the length of the zip code field in the LENGTH field. The length must be the field length as specified in the panel definition.
- Enter your test input to the field edit in the INPUT field.
- Press Enter.



As shown below, the Field Edit Test Menu refreshes with the output. The zip code output is **03049** because that is a valid zip code according to the edit. The return code of 0000 indicates that the edit worked.

```
FIELD EDIT TEST MENU ********************* ********* EDIT PROGRAM RETURN CODE - 0000
COMMAND ==> ____________________________________________________________________

 FIELD EDIT NAME IZIPA___   EDIT TYPE A (A/N/B/V/L)    WORKFLD NUMERIC __ V __

 STOP AT PROGRAM INIT  N

---------------------------- STANDARD PARAMETERS ----------------------------
 LENGTH:   5 (TPO/TPI)
        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
 INPUT:  03049_____________________________________________________________
   *     _____________________________________________________________
 OUTPUT: 03049
   *
 ERROR:

---------------------------- EXTENDED PARAMETERS --------------------- PAGE 01
 NBR TYPE                 VALUE/DBDNAME/PCB NUMBER                    NUMERIC
  1  ___ ____________________________________________________________   __ V __
  2  ___ ____________________________________________________________   __ V __
  3  ___ ____________________________________________________________   __ V __
  4  ___ ____________________________________________________________   __ V __
  5  ___ ____________________________________________________________   __ V __
```

At this point, you can:

- Test more field edits
- Exit

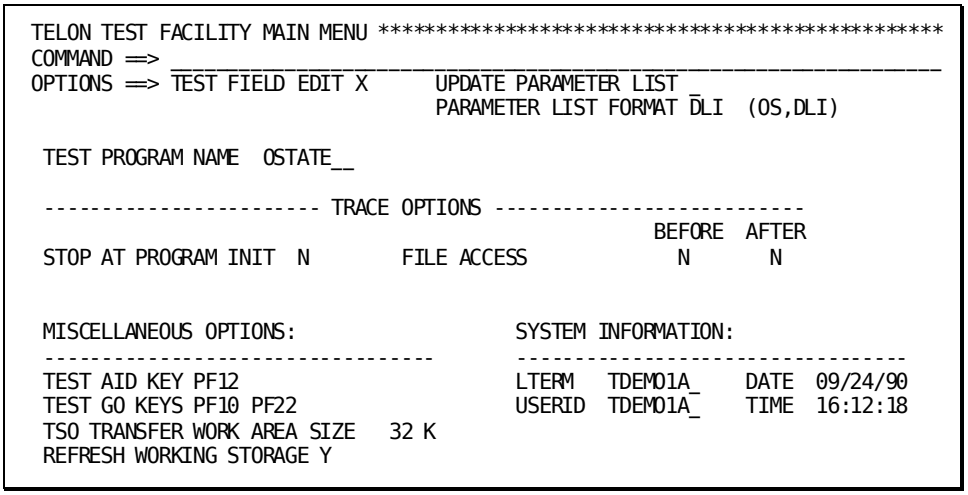
If you want to test more field edits, Enter the next field edit in the FIELD EDIT NAME field on the Field Edit Test Menu and press Enter.

Output Field Edit

An output field edit reformats data for display on the application screen. The output field edit shown in this example, OSTATE, converts a state's two-letter postal abbreviation to the full name of the state when it displays the state field during runs of the application.

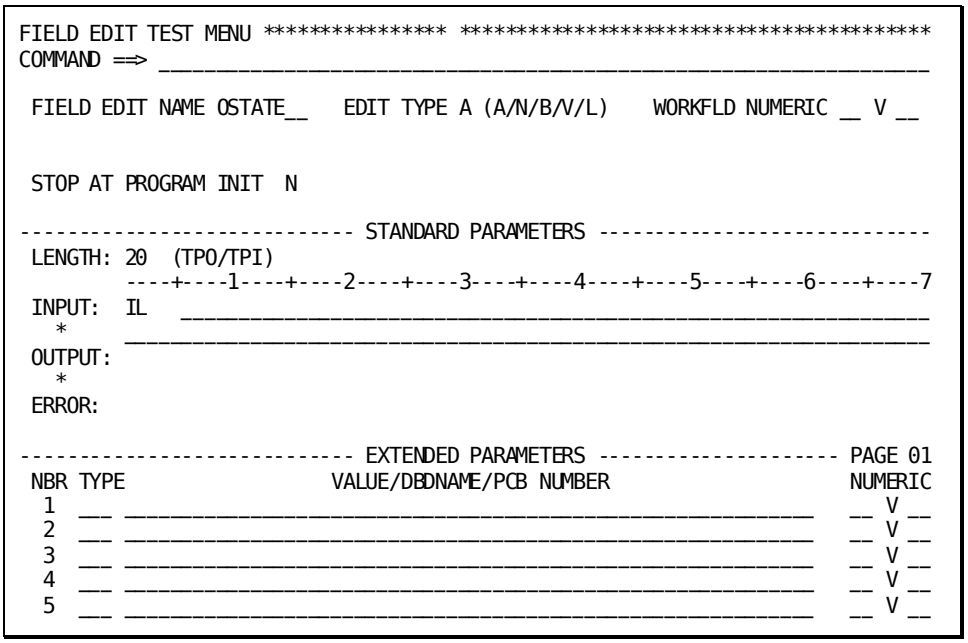
As shown below on the CA Telon Test Facility Main Menu:

- Enter **X** in the TEST FIELD EDIT field
- Enter the name of the field edit in the TEST PROGRAM NAME field
- Press Enter.



As shown in the example below, the TTF returns the Field Edit Test Menu. On this menu:

- Enter **A** for alpha in the EDIT TYPE field.
- Enter the length of the state field in the LENGTH field. The length must be the field length as specified in the panel definition.
- Enter your test input to the field edit in the INPUT field.
- Press Enter.



As illustrated below, the Field Edit Test Menu refreshes with ILLINOIS in the OUTPUT field as shown in the next screen. The return code is 0000, so the edit worked. The subroutine connected the correct state name to the abbreviation.

```
FIELD EDIT TEST MENU ********** ******** EDIT PROGRAM RETURN CODE - 0000
COMMAND ==> _____________________________________________________________

 FIELD EDIT NAME OSTATE__   EDIT TYPE A (A/N/B/V/L)    WORKFLD NUMERIC __ V __

 STOP AT PROGRAM INIT  N

---------------------------- STANDARD PARAMETERS ----------------------------
 LENGTH:  20 (TPO/TPI)
        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
 INPUT:  IL_____________________________________________________________
   *     _____________________________________________________________
 OUTPUT: ILLINOIS
   *
 ERROR:

---------------------------- EXTENDED PARAMETERS --------------------- PAGE 01
 NBR TYPE                 VALUE/DBDNAME/PCB NUMBER                    NUMERIC
  1  ___ ____________________________________________________________   __ V __
  2  ___ ____________________________________________________________   __ V __
  3  ___ ____________________________________________________________   __ V __
  4  ___ ____________________________________________________________   __ V __
  5  ___ ____________________________________________________________   __ V __
```

At this point, you can:

- Test more field edits
- Exit

To test more field edits, Enter the name of the next field edit in the FIELD EDIT NAME field and press Enter.

Chapter 4: Prototyping

This chapter discusses three different kinds of prototyping or screen execution with the CA Telon Test Facility.

As outlined earlier, the types of prototyping and testing available to the CA Telon application developer are:

- Prototyping without data mapping
- Prototyping with data mapping
- Screen execution without data access
- Screen execution with generic data access
- Screen execution with production databases

See the *Programming Concepts Guide* for an explanation of the first two levels of prototyping. The chapter on the prototyping facility contains step-by-step examples and describes all of the available features and parameters.

Prototyping with the Test Facility

Screen execution in the TTF allows you to view how your applications will look in production.

To prototype under the TTF, you must perform the following steps:

1. Create the program definition
2. Generate and compile the program
3. Execute under the TTF

You can combine different types of screen execution under the TTF when prototyping. The type of screen execution you use depends on the databases available at the time and what you want to demonstrate with the prototype. Data for existing databases can be accessed the same as for the final application, while data without existing databases can be mapped in and out of the Transfer Work Area.

Screen Execution without Data Access

This level of prototyping is best used to demonstrate that a concept works. You can design a mini application to prove that a proposed concept is functional. Although screen execution without data access requires frequent programmer intervention, you can demonstrate the editing capabilities of an application and test generated loop logic.

After you have prototyped panel definitions with the prototyping facility, screen execution without data access makes the prototype more realistic.

This means that screen flow occurs automatically, and you can include any additional functions such as PF keys, help/hold processing, and field and consistency edits.

You do not need databases at this level of prototyping. You simulate data access by mapping data in and out of the Transfer Work Area. In other words, you use Transfer Work Area Names as DBnames in the panel definition.

Transfer Work Area Use

Logically, the mapping of data here is similar to mapping in the prototyping facility. In this case, however, the *Transfer Work Area* maps data instead of the Active Presentation Store. The entry and display of data appears the same as for Prototyping, except that the Undefined Store Character (defined on the Prototyping Facility Menu) is not used.

To enable that mapping of data to and from the Transfer Work Area, you must add the field names (DBNAMEs from the panel definition) to the COPY book for the Transfer Work Area. If initial segment definitions have been determined at this point (either with or without a data dictionary), the easiest way is simply to copy the COPY book entries for all segments into the COPY book entry for the Transfer Work Area.

List Screens

Screen execution without data access does not simulate multiple values on a List screen. This technique can only list a single occurrence of a group unless data is stored within an array in the Transfer Work Area. If an application includes a significant number of list screens, use screen execution with Generic Data Access.

However, if you set PGMINIT to **Y**, you will have a break point before every iteration of SEGLOOP processing. This would allow you to plug in new data each time the program stops until you reach the end of the loop.

Screen Execution with Generic Data Access

This level of prototyping allows you to verify all of the functions of your program. You make the prototype more realistic by accessing generic databases or files. In this way, you avoid designing databases until you have completed most of your application. Screen execution with generic data access allows you to prove that the entire system works.

The databases or files defined in those screen definitions are general-purpose *generic* databases or files, instead of those that are actually used in the final application. The data items created for each data group define databases which are not a part of the final application, but exist only for prototyping. The prototype appears to the application user the same as the final application with data entered, saved, displayed, and listed. In terms of TTF function, there is no difference between this level and prototyping with the application databases.

Generic Databases

Your installation should have a few generic databases defined for prototyping use.

When you run the TTF each generic database definition can be used by more than one application prototype at the same time. This is because different filenames can be defined in each CLIST used.

If multiple applications use a database, each application must have both of the following:

- Unique segment layouts (COPY books) within the copy library
- Unique data set names within the TTF

You can create unique layouts in one of the following ways:

- Using unique copy libraries for each application.
- Overriding the COPY book name for the segment in the screen definition. This is easier if you create a screen definition base for the application, with the segments set up properly in the base.

You can define unique data set names by specifying the names in the TTF CLIST for the application.

Recommendations for Generic Databases

When setting up generic databases, you should:

- Give the databases simple segment names that are easy to remember, such as DB1SEGA and DB2SEGB.
- Give search field names (key names in the DBD) simple names, similar to the segment names.
- Create each database structure to include multiple segment types at each level, down to about three levels.
- Create at least some of the structures with secondary indexes at the first and second levels.
- Make segments large enough to support segment layouts required by the application prototypes.
- Make segment key fields large enough to support the keys of the application models. There is no problem if the key in the segment is larger than the key in the application. COBOL and PL/I fill the segment key with spaces when the segment is written and when the SSA for the segment is set up.

Usually, it is easier to avoid creating prototyping and generic data structures that are unique to a particular application. However, if your installation can generate temporary database structures easily, you can use an interim database design for the prototype, knowing that the database will be modified before the application goes into production. In this case, you can use much of the data definition within the application prototype, without further modification, when carrying the model into the final application definition stage.

You can also use prototyping with generic data structures to debug application coding, when the application programming tasks would otherwise have to wait for the completion of database definition. This way, you can address many of the programming tasks in parallel with database design.

Conversion to Production Databases

When the final application databases are ready, the effort required to convert each screen definition to the final database definitions depends on:

- The similarity of the new databases to the generic databases
- The type of I/O used (Auto Exec or User Exec)

It is often easier to delete all data items defined in a screen definition and redefine them from scratch, rather than modifying the prototype data item entries. If the prototype used Auto Exec, this is usually all that is required. If the prototype used User Exec, you must modify each custom code reference to a segment so that it accesses the production database segment(s). Key length differences and search names are usually transparent in the conversion to the new databases.

Screen Execution with Production Databases

The final prototype level accesses the actual production databases. The DBD definitions used are the current definitions of those databases. This does not, however, mean that you cannot modify the database prior to production. Screen execution with production databases is the same as regular system testing.

This prototype can be functionally very close to the final production system. Although some edits might be missing and security and audit trails might not be implemented, from an end-user perspective the prototype functions the same as the final production application. The major difference is that you can run the prototype run under the test facility (for IMS/DC installations, within TSO instead of IMS/DC). In this mode, the application user actually tests the system by executing it against real data in real-life scenarios. The user's testing determines if the application effectively performs the required functions.

Prototyping or Testing

At this level, you can consider the application user involvement as either prototyping or testing. In fact, no clear line exists between prototyping and testing. Prototyping is simply the testing of early design decisions, and in the evolution through general design, detail design, programming and system testing, the distinctions melt away.

Application Completion

Before you begin real data access prototyping, be sure that the programs developed so far effectively perform the application function.

After prototyping with real data access, finish the development of the application with the final:

- CA Telon coding
- Generation into the target environment
- System test of the application

In the final system test, you must assure proper production execution, verifying that:

- All edits are complete
- The production environment is set up correctly
- Audit trails and security are implemented correctly
- Performance criteria are met
- Functions still perform correctly within the production environment at production volumes

Chapter 5: Sample Session

This chapter follows an online CA Telon application through the testing process within the CA Telon Test Facility. This chapter is intended to provide you with a basis for testing with the CA Telon Test Facility. After following this example, you should understand the basic functions of the TTF and how to test your own programs.

The example provided in this chapter shows data access for DL/I and DB2. Although both data accesses are illustrated here, you might have **only** one type of data access per program.

You would use the Test Facility to test CA Telon TSO, IMS/DC, and Batch programs. For CA Telon TSO programs, the TTF simulates execution in an IMS/DC environment without requiring IMS resources. The Test Facility also allows you to test field edits or routines.

Prerequisites to Testing

For the test facility to generate trace tables, you must instruct the CA Telon TDF to generate trace logic in the application program. Specify **Y** in the TRACE field on the TDF Environment screen. The Trace Tables are extremely helpful later in the debugging process. The *Programming Concepts Guide* and the *Design Facility Reference guide* both contain additional information on the TDF Environment screen.

Before you can test a program, you must:

- Generate it so CA Telon source statements generate a COBOL or PL/I source program.
- Compile it to create a temporary object module.
- Link it into a load library so you can test it. Load module names vary depending on installation standards.

Test Facility Example

This section provides an example of the CA Telon Test Facility. This example demonstrates a sample debugging session, explains what to do, and discusses what the Test Facility does for you.

Note: This example uses VSAM files, but it also shows data access for DB2 and DL/I.

The structure of the test program determines the flow through the TTF. The different Update and Display screens that appear are a result of the program flow specified in the actual program you are testing.

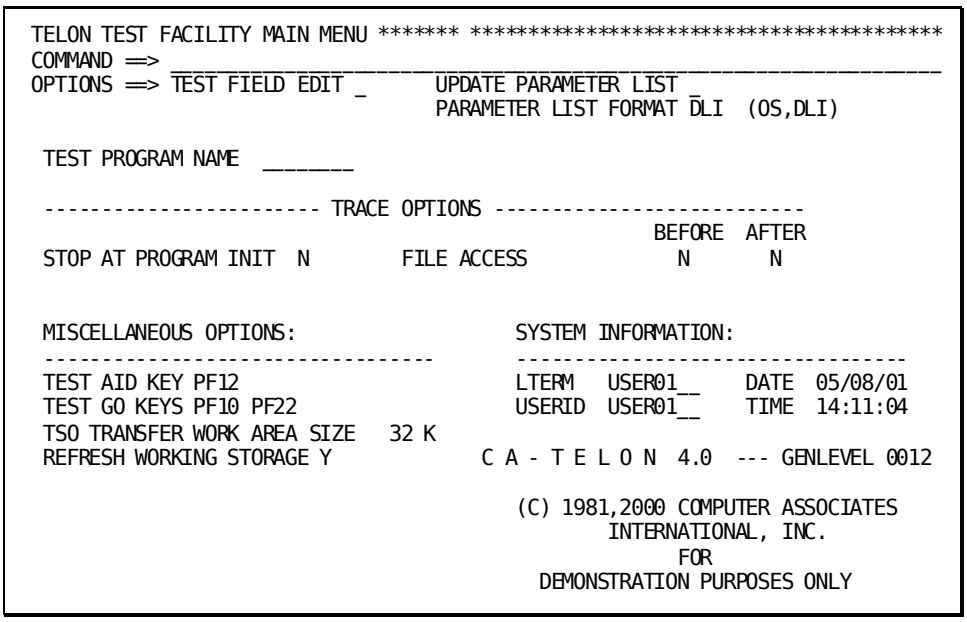
Access the Test Facility

To access the Test Facility:

- Logon to TSO
- In ISPF Option 6 execute the TELON CLIST @TLTFTD
- For DB2 programs execute CLIST @TLTFT2

From the CA Telon Test Facility Main Menu you can:

- Enter the load module name in the TEST PROGRAM NAME field
- Set trace options, if desired



Break Points

By alternately stopping and resuming execution of a program under the CA Telon Test Facility, you make sure that it functions correctly and that it meets all requirements. You do this by checking the status of database and file accesses, examining the contents of storage and checking that data transfers correctly from screen to screen.

To stop execution of a program, you set break points. It is simple to set break points at program initiation, and before and after database or file accesses. Specify **Y** for File Access Before and After and **Y** for Program Init in the Trace options. You can also set custom break points using the COMMAND field. See Chapter 3, "Advanced Procedures," earlier in this manual, for more detailed information about custom break points.

TEST GO Key

To resume execution of a program, press the TEST GO key or enter **GO** on the COMMAND field. The installation default for the TEST GO key is **PF10**/**PF22**. You can alter this on the TTF Main Menu.

TEST AID Key

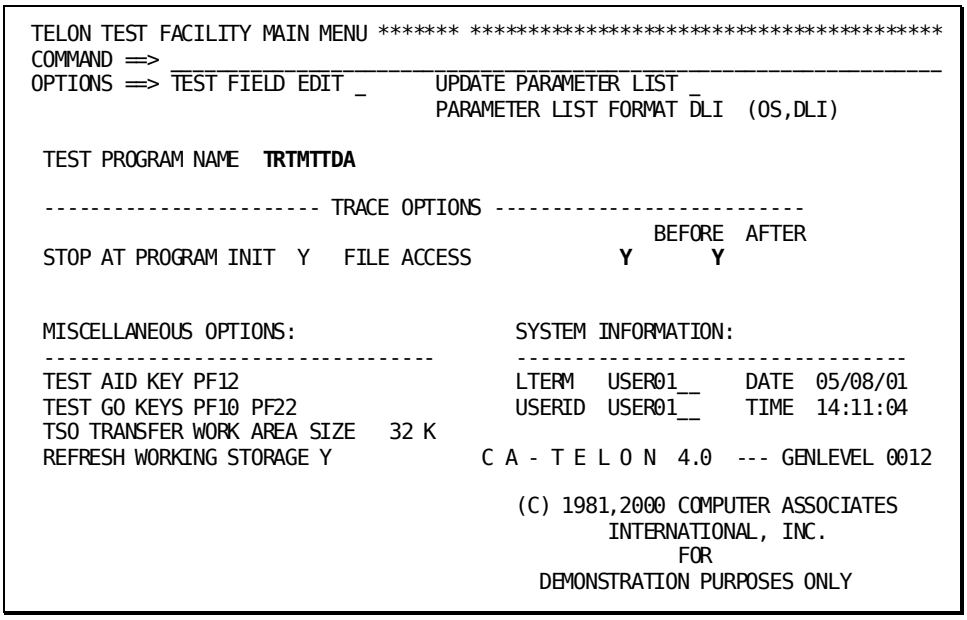
To inspect storage or set break points, press the TEST AID key to toggle to a Debug Application screen. The installation default for the TEST AID key is **PF12**. You can alter this on the TTF Main Menu.

From the Debug Application screen, you can inspect the contents of storage and perform other test facility functions.

Command Field

All Test Facility screens have a COMMAND field. It is not always in the same place on every screen, but it is always present. See the "Commands" appendix for a complete list of commands and more information.

As shown below, your logical terminal ID and userid are automatically displayed on the screen.



Press Enter to proceed.

CA Telon now displays the Debug Application screen because the trace option to stop at program initiation is set to **Y**.

The top portion of the screen displays general information about the program tested, including the ID and address of specific areas of storage.

The section name where the program has stopped is displayed in the middle of the screen. A window of unprotected storage appears directly below it. Section Q-100 is the program initiation point. Press **PF7** and **PF8** to scroll through the contents of storage.

```
DEBUG APPLICATION ****************** ****************************************
 PROGRAM  ENTRY POINT      DATE        TIME     LENGTH    PCB LIST  TRACE TABLE
TRTMTTDA   00151450     02 02 2001   14 14 59   007BB0    80054054   000B6010

                ID      ADDR          ID      ADDR          ID      ADDR
 TRACE     01 XFERWORK 0015A010   05 PGM WORK 001A8988
 TABLE:    02 WRK STRG 001A8088   06 SSA AREA 001A8730
   *       03 SYS WORK 001A89F0   07 TRGEMPL  001A8CBC
   *       04 APP WORK 001A8848
-----------------------------------  ---------------  ------------------------
    FUNCTION                          SECTION  FIELD        MODULE         DISP
  PROGRAM INIT                        Q-100              TRTMTTDA.TRTMTTDA 000000
00151450   * 47F0F028  00C3C5C5   00000218  00000014 *   * .00..CEE........ *
00151460   * 47F0F001  98CEAC00   00151506  00000000 *   * .00............. *
00151470   * 00000000  00000000   90ECD00C  4110F038 *   * .............0. *
00151480   * 98EFF04C  07FF0000   00151450  00000000 *   * ..0<.......&.... *
00151490   * 00156A80  001514FE   00151450  001533F2 *   * ...........&...2 *

 TRACE:    PGM INIT Y   FILE BEFORE Y - AFTER Y

COMMAND ==> _____________________________________________ SCROLL ==> PAGE
  AREA:     TRACE ID  _____________  LINE __        DISPLAY LTH  ___
   *        STOP AT ID _______                      UPDATE C/X _
```

Modify the Contents of Storage

To modify the contents of storage:

- Enter the number or address of the Trace Table item which points to the location in storage you want to examine in the COMMAND field.

  On this screen, **1** requests the XFERWORK area. The Transfer Work Area contains data to be shared by different programs in the application.

- Press Enter.

The requested XFERWORK information now appears in the mid-lower portion of the screen. It is positioned at the starting address of the area now displayed. These addresses appear in bold for illustration.

As shown below, the screen allows you to modify the contents of storage by typing over the data in hex or character format.

```
DEBUG APPLICATION ****************** ****************************************
 PROGRAM  ENTRY POINT      DATE        TIME     LENGTH    PCB LIST  TRACE TABLE
TRTMTTDA   00151450     02 02 2001   14 14 59   007BB0    80054054   000B6010

               ID      ADDR           ID      ADDR           ID      ADDR
 TRACE     01 XFERWORK 0015A010   05 PGM WORK 001A8988
 TABLE:    02 WRK STRG 001A8088   06 SSA AREA 001A8730
   *       03 SYS WORK 001A89F0   07 TRGEMPL  001A8CBC
   *       04 APP WORK 001A8848
---------------------------------  ---------------  ------------------------
    FUNCTION                         SECTION  FIELD        MODULE         DISP
  PROGRAM INIT                       Q-100                  UNKNOWN
0015A010   * 00000000  00000000   00000000  00000000 *   * 123456    ...... *
0015A020   * 00000000  00000000   00000000  00000000 *   * 00010508 ....... *
0015A030   * 00000000  00000000   00000000  00000000 *   * ................ *
0015A040   * 00000000  00000000   00000000  00000000 *   * ................ *
0015A050   * 00000000  00000000   00000000  00000000 *   * ................ *

 TRACE:    PGM INIT Y   FILE BEFORE Y - AFTER Y

COMMAND ==> _____________________________________________ SCROLL ==> PAGE
  AREA:     TRACE ID  _____________  LINE __        DISPLAY LTH  ___
   *        STOP AT ID _______                      UPDATE C/X C
```

You must supply the following data in the Transfer Work Area that will be displayed on the screen when the program finishes output processing:

- Enter the employee number followed by four spaces
- Enter "oo", followed by the date (YYMMDD), followed by a space
- Enter **C** in the UPDATE C/X field to write the characters to storage
- Press Enter
- Press PF8 to page down
- Enter "F"
- Enter "X" in the Update C/X field to write the hex value to storage

- Press Enter.

```
DEBUG APPLICATION ****************** ****************************************
 PROGRAM  ENTRY POINT      DATE        TIME     LENGTH    PCB LIST  TRACE TABLE
TRTMTTDA   00151450     02 02 2001   14 14 59   007BB0    80054054   000B6010

               ID      ADDR           ID      ADDR           ID      ADDR
 TRACE     01 XFERWORK 0015A010   05 PGM WORK 001A8988
 TABLE:    02 WRK STRG 001A8088   06 SSA AREA 001A8730
   *       03 SYS WORK 001A89F0   07 TRGEMPL  001A8CBC
   *       04 APP WORK 001A8848
----------------------------------  ---------------  ------------------------
    FUNCTION                        SECTION  FIELD       MODULE         DISP
  PROGRAM INIT                      Q-100                UNKNOWN
0015A060   * 00000000  00000000   00000000  00000000 *   * ................ *
0015A070   * 00000000  00000000   00000000  0f000000 *   * ................ *
0015A080   * 00000000  00000000   00000000  00000000 *   * ................ *
0015A090   * 00000000  00000000   00000000  00000000 *   * ................ *
0015A0A0   * 00000000  00000000   00000000  00000000 *   * ................ *

 TRACE:    PGM INIT Y   FILE BEFORE Y - AFTER Y

COMMAND ==> _____________________________________________ SCROLL ==> PAGE
  AREA:     TRACE ID  _____________   LINE __        DISPLAY LTH  ___
   *        STOP AT ID _______                        UPDATE C/X X
```

- Press PF8 again to page forward
- Enter the characters shown
- Enter **c** in the Update C/X field to write the characters to storage

```
DEBUG APPLICATION ****************** ****************************************
 PROGRAM  ENTRY POINT      DATE        TIME     LENGTH    PCB LIST  TRACE TABLE
TRTMTTDA   00151450     02 02 2001   14 14 59   007BB0    80054054   000B6010

               ID      ADDR           ID      ADDR           ID      ADDR
 TRACE     01 XFERWORK 0015A010   05 PGM WORK 001A8988
 TABLE:    02 WRK STRG 001A8088   06 SSA AREA 001A8730
   *       03 SYS WORK 001A89F0   07 TRGEMPL  001A8CBC
   *       04 APP WORK 001A8848
----------------------------------  ---------------  ------------------------
    FUNCTION                        SECTION  FIELD       MODULE         DISP
  PROGRAM INIT                      Q-100                UNKNOWN
0015A0B0   * 00000000  00000000   00000000  00000000 *   * ................ *
0015A0C0   * 00000000  00000000   00000000  00000000 *   * .............TTD *
0015A0D0   * 00000000  00000000   00000000  00000000 *   * M                *
0015A0E0   * 00000000  00000000   00000000  00000000 *   *  01001001....... *
0015A0F0   * 00000000  00000000   00000000  00000000 *   * ................ *

 TRACE:    PGM INIT Y   FILE BEFORE Y - AFTER Y

COMMAND ==> _____________________________________________ SCROLL ==> PAGE
  AREA:     TRACE ID  _____________   LINE __        DISPLAY LTH  ___
   *        STOP AT ID _______                        UPDATE C/X C
```

Press the TEST GO key to resume program execution.

DL/I Data Access

As shown in the example below, the TTF accesses the Trace DL/I Access Before screen as a result of the Trace Option settings.

This screen displays:

- The SECTION performed in the program
- The dataset (DDname) accessed
- The function performed

For the purposes of this example, the headings appear in bold. The RECORD ID field displays an employee number, **123456**, which was entered and the previous screen and passed to this part of the program through the Transfer Work Area.

If you want to:

- Alter the trace options on subsequent calls, enter **Y** or **N** in the TRACE BEFORE and/or TRACE AFTER fields in the lower right portion of the screen
- Cancel this call, enter **Y** in the CANCEL CALL field, shown in the lower right portion, and press Enter.

```
TRACE DL/I ACCESS BEFORE ************ **** ------ PCB INFO ------ *************
 PROGRAM   SECTION    DBD      PCB  FUNC   STATUS  SEGMENT  LEVEL      DLI PARM
 TRTMTTDA  A-100    TRGDBDV1   006  GU                                 001841F0

 NO     ADDR    ----- S S A   L I S T -----
  1   801A873C  TRGEMPL *---(TRGEMKEY= 123456)




LENGTH  ADDR     - K E Y   F E E D B A C K   A R E A -     -- CHAR FORMAT ---
                                                         *               *
                                                         *               *
LENGTH  ADDR     ---------- I/O   A R E A ------------     -- CHAR FORMAT ---
      001A8CBC   00000000 00000000    00000000 00000000    *...............*
      001A8CCC   00000000 00000000    00000000 00000000    *...............*
      001A8CDC   00000000 00000000    00000000 00000000    *...............*
      001A8CEC   00000000 00000000    00000000 00000000    *...............*
      001A8CFC   00000000 00000000    00000000 00000000    *...............*

COMMAND ==> _____________________________________ TRACE BEFORE Y - AFTER Y
OPTIONS ==> UPDATE DLI TRACE _                    CANCEL CALL  _
```

Press the TEST GO key to resume program execution.

As illustrated in the example below, the Trace DL/I Access After screen displays the result of the file access.

The RESPONSE field contains the return code or status of the file access. The record is retrieved into the I/O area and displayed in hex and character format.

The information read from the database is mapped to variables that are part of the screen panel definition in output processing and displayed on the screen.

```
TRACE DL/I ACCESS AFTER  ********** **** ------ PCB INFO ------ **********
 PROGRAM   SECTION    DBD      PCB  FUNC   STATUS  SEGMENT  LEVEL      DLI PARM
 TRTMTTDA  A-100    TRGDBDV1   006  GU             TRGEMPL   01        001841F0

 NO     ADDR    ----- S S A   L I S T -----
  1   801A873C  TRGEMPL *---(TRGEMKEY= 123456)




LENGTH  ADDR     - K E Y   F E E D B A C K   A R E A -     -- CHAR FORMAT ---
   6  000AA5AC   F1F2F3F4 F5F6                            *123456          *
                                                         *                *
LENGTH  ADDR     ----------- I/O   A R E A ------------    -- CHAR FORMAT ---
      001A8CBC   F1F2F3F4 F5F6C2C9    D9C46B40 D3C1D9D9    *123456BIRD, LARR*
      001A8CCC   E8404040 40404040    40404040 404040F5    *Y              5*
      001A8CDC   F8F1F0F2 F4D4F1F1    F3F5F5F5 F1F2F1F2    *81024M1135551212*
      001A8CEC   D6D5C540 C3C5D3E3    C9C340D3 C1D5C540    *ONE CELTIC LANE *
      001A8CFC   40404040 40404040    40C6D9C5 D5C3C840    *         FRENCH *

COMMAND ==> ____________________________________ TRACE BEFORE Y - AFTER Y
OPTIONS ==> UPDATE DLI TRACE _
```

Press the TEST GO key to resume program execution.

DB2 Data Access

The example below shows the TTF screens for a CA Telon program that accesses a DB2 table for processing.

The program section is A-100. The COMMAND is SELECT and there are 14 columns in the table to be selected.

```
TRACE DB2 ACCESS BEFORE ************ *************************** PAGE 01 ****
 PROGRAM  SECTION          COMMAND   SQLCODE   COLCNT                 DB2 PARM
 TRTMTT2A A-100            SELECT               014                  001540A8

COL LTH   INTO    --------- I/O   A R E A S ----------  -- CHAR FORMAT --- TYPE
001 006 00152900  F1F2F3F4 F5F6                        *123456          * CHR
002 006 00152868  00000000 0000                        *......          * CHR
003 025 0015286E  00000000 00000000  00000000 00000000  *................* CHR
004 004 00152887  00000000                             *....            * DEC
005 001 0015288B  00                                   *.               * CHR
006 010 0015288C  00000000 00000000  0000              *..........      * CHR
007 025 00152896  00000000 00000000  00000000 00000000  *................* CHR
008 025 001528AF  00000000 00000000  00000000 00000000  *................* CHR
009 002 001528C8  0000                                 *..              * CHR
010 005 001528CA  00000000 00                          *.....           * CHR
011 004 001528CF  00000000                             *....            * DEC
012 003 001528D3  000000                               *...             * CHR
013 003 001528D6  000000                               *...             * DEC
014 003 001528D9  000000                               *...             * DEC


COMMAND ==> ____________________________________________ TRACE BEFORE Y - AFTER Y
OPTIONS ==> UPDATE DB2 TRACE _                          CANCEL CALL  _
```

Press the TEST GO key to resume program execution.

As shown below, the TTF accesses the Trace DB2 Access After screen which displays the SQLCODE and the columns of data returned from the call.

```
TRACE DB2 ACCESS AFTER  ************ *************************** PAGE 01 ****
 PROGRAM  SECTION          COMMAND   SQLCODE   COLCNT                  DB2 PARM
 TRTMTT2A A-100            SELECT     +000      014                    001540A8

COL LTH   INTO    --------- I/O   A R E A S ----------  -- CHAR FORMAT --- TYPE
001 006 00152900  F1F2F3F4 F5F6                        *123456          * CHR
002 006 00152868  F1F2F3F4 F5F6                        *123456          * CHR
003 025 0015286E  C2C9D9C4 6B40D3C1  D9D9E840 40404040 *BIRD, LARRY     * CHR
004 004 00152887  0581124C                             *...>            * DEC
005 001 0015288B  D4                                   *M               * CHR
006 010 0015288C  F4F1F3C6 D3F9F7F7  F6F6              *413FL97766      * CHR
007 025 00152896  D6D5C540 C3C5D3E3  C9C340D3 C1D5C540 *ONE CELTIC LANE * CHR
008 025 001528AF  C6D9C5D5 C3C840D3  C9C3D240 40404040 *FRENCH LICK     * CHR
009 002 001528C8  D6C8                                 *OH              * CHR
010 005 001528CA  F4F7F5F7 F8                          *47578           * CHR
011 004 001528CF  0840831C                             *. ..            * DEC
012 003 001528D3  C2C1E2                               *BAS             * CHR
013 003 001528D6  02000C                               *...             * DEC
014 003 001528D9  00230C                               *...             * DEC


COMMAND ==> _____________________________________ TRACE BEFORE Y - AFTER Y
OPTIONS ==> UPDATE DB2 TRACE _
```

Press the TEST GO key to resume program execution.

The TTF displays the Employee Add/Update screen, shown below, until you enter information.

Employee number 123456, previously entered in the Transfer Work Area, serves as the key to the database. Using this employee number key, the program retrieved and now displays information for the employee.

```
05/08/01        T E L O N   S A M P L E   S O L U T I O N
                          EMPLOYEE UPDATE

                        EMPLOYEE ID  123456


 1. NAME                BIRD, LARRY_____________
 2. STREET              ONE CELTIC LANE_________
 3. CITY                FRENCH LICK_____________
 4. STATE               IN
 5. ZIP                 47578
 6. PHONE               413-FL9-7766
 7. SEX                 M
 8. DATE OF BIRTH       10/24/58
 9. DEPARTMENT          BAS
10. DATE OF EMPLOYMENT  08/31/84
11. HOURLY RATE          20.00
12. HOURS PER WEEK        23




_____________________________________________________________________________
 PFKEYS:  2-HOLD  3-END  4-ENDHOLD  5-CANCEL  6-MENU
```

Update the record by overtyping the displayed information, as shown in the Employee Add/Update screen below.

```
05/08/01        T E L O N   S A M P L E   S O L U T I O N
                          EMPLOYEE UPDATE

                        EMPLOYEE ID  123456


 1. NAME                BIRD, LARRY_____________
 2. STREET              ONE FLEETCENTER WAY_____
 3. CITY                BOSTON
 4. STATE               MA
 5. ZIP                 02101
 6. PHONE               617-495-3333
 7. SEX                 M
 8. DATE OF BIRTH       10/24/58
 9. DEPARTMENT          BAS
10. DATE OF EMPLOYMENT  08/31/84
11. HOURLY RATE          20.00
12. HOURS PER WEEK        23




_____________________________________________________________________________
 PFKEYS:  2-HOLD  3-END  4-ENDHOLD  5-CANCEL  6-MENU
```

Press Enter to invoke input processing in the application program.

Display Full Screen Storage

In the next example below, the TTF accesses the Debug Application screen.

```
DEBUG APPLICATION ****************** ****************************************
 PROGRAM  ENTRY POINT      DATE        TIME     LENGTH    PCB LIST  TRACE TABLE
TRTMTTDA   00151450     02 02 2001   14 14 59   007BB0    80054054   000AA010

               ID      ADDR          ID      ADDR          ID      ADDR
 TRACE     01 XFERWORK 0015A010   05 PGM WORK 001A8988
 TABLE:    02 WRK STRG 001A8088   06 SSA AREA 001A8730
   *       03 SYS WORK 001A89F0   07 TRGEMPL  001A8CBC
   *       04 APP WORK 001A8848
---------------------------------  ---------------  ------------------------
    FUNCTION                        SECTION  FIELD       MODULE         DISP
  PROGRAM INIT                      Q-100            TRTMTTDA.TRTMTTDA 000000
00151450   * 47F0F028  00C3C5C5   00000218  00000014 *   * .00..CEE........ *
00151460   * 47F0F001  98CEAC00   00151506  00000000 *   * .00............. *
00151470   * 00000000  00000000   90ECD00C  4110F038 *   * ..............0. *
00151480   * 98EFF04C  07FF0000   00151450  00000000 *   * ..0<.......&.... *
00151490   * 00156A80  001514FE   00151450  001533F2 *   * ...........&...2 *

 TRACE:    PGM INIT Y   FILE BEFORE Y - AFTER Y

COMMAND ==> DISPLAY S_____________________________________ SCROLL ==> PAGE
  AREA:     TRACE ID  ______________  LINE __        DISPLAY LTH  ___
   *        STOP AT ID _______                       UPDATE C/X _
```

To view storage on a full-screen basis:

- Enter **DISPLAY S** on the COMMAND field
- Press Enter

COMMAND

This field allows you to enter any of the valid test commands activated for the current test screen. See the "Commands" appendix for a complete list of commands and more information.

On the Display Full Screen Storage screen below, you can scroll through the storage displayed by pressing **PF7** (scroll backward) and **PF8** (scroll forward). You can also limit the length of storage displayed to a specific number of bytes.

```
DISPLAY FULL SCREEN STORAGE ********* ************************************
COMMAND ==> _____________________________________________ SCROLL ==> PAGE
  AREA:     UPDATE C/X _   DISPLAY LTH ___                MODULE         DISP
                                                    TRTMTTDA.TRTMTTDA 000000
 00151450   * 47F0F028  00C3C5C5  00000218  00000014 *   * .00..CEE........ *
 00151460   * 47F0F001  98CEAC00  00151506  00000000 *   * .00.............. *
 00151470   * 00000000  00000000  90ECD00C  4110F038 *   * ..............0. *
 00151480   * 98EFF04C  07FF0000  00151450  00000000 *   * ..0<.......&.... *
 00151490   * 00156A80  001514FE  00151450  001533F2 *   * ..........&...2 *
 001514A0   * 00158598  0015151A  00104001  00000008 *   * .......... ..... *
 001514B0   * E3D9E3D4  E3E3C4C1  F2F0F0F1  F0F2F0F2 *   * TRTMTTDA20010202 *
 001514C0   * F1F4F1F4  F5F9F0F1  F0F2F0F0  00000000 *   * 141459010200.... *
 001514D0   * 00000000  C0E86E4C  00000000  50000B29 *   * .....Y><....&... *
 001514E0   * 01080000  08000000  00800000  000002EE *   * ................ *
 001514F0   * 000002A8  00008000  40404040  0008E3D9 *   * ........    ..TR *
 00151500   * E3D4E3E3  C4C10500  00010015  85180000 *   * TMTTDA.......... *
 00151510   * 0000FFFF  FFB20015  14500015  14880000 *   * .........&...... *
 00151520   * 00080000  00060015  14880015  85180015 *   * ................ *
 00151530   * 84F00000  00050000  00000000  00000000 *   * .0.............. *
 00151540   * 00000000  00000000  00000000  00000001 *   * ................ *
 00151550   * 40404040  40404040  40404040  40404040 *   *                  *
 00151560   * 40404040  40404040  40404040  4040F0F0 *   *                00 *
 00151570   * F0F0F0F0  F0F0F0F0  F0F0F0F0  F0F0F0C0 *   * 000000000000000. *
```

The example below illustrates pressing **PF8** to scroll down on the Display Full Screen Storage screen.

```
DISPLAY FULL SCREEN STORAGE ********* ****************************************
COMMAND ==> _____________________________________________ SCROLL ==> PAGE
  AREA:     UPDATE C/X _   DISPLAY LTH ___              MODULE         DISP
                                                    TRTMTTDA.TRTMTTDA 000130
 00151580   * 00000000  0F404040   001515A4  001525A4 *   * .....   ........ *
 00151590   * 00152908  00153924   00154932  00155968 *   * ................ *
 001515A0   * 00156978  00157160   00158238  00157030 *   * .......-........ *
 001515B0   * 00158120  00158088   00157F08  00157EE0 *   * ..............=. *
 001515C0   * 00157DB0  00157CD8   00157B48  00157820 *   * .......Q........ *
 001515D0   * 001576A0  001574E0   001574A0  00156D88 *   * ..............._. *
 001515E0   * 0015294A  001533F2   0015618A  00153696 *   * .......2........ *
 001515F0   * 00153730  00153FB4   0015382C  0015408C *   * .............. . *
 00151600   * 00156A22  00154CA2   00153A90  00153D5C *   * ......<........* *
 00151610   * 00155FE6  00154184   00154804  0015676E *   * ...W..........> *
 00151620   * 001549D0  00156698   001569A4  00155A9E *   * ................ *
 00151630   * 00156308  00155A08   001565E6  00156480 *   * ...........W.... *
 00151640   * 00155282  00154EA0   00001000  00000050 *   * ......+........& *
 00151650   * FFFFFFF8  40000258   00000014  FFFFFFFC *   * ...8 ........... *
 00151660   * 40000004  00000048   00000001  0000C0C5 *   *  ..............E *
 00151670   * 0000C0C9  0000C0C1   00000000  0DB50DB4 *   * ...I...A........ *
 00151680   * 0DB30DB2  0DB10DB0   0DAF0DAE  0DAD0DAC *   * ................ *
 00151690   * 0FA10FA2  0090003F   0103000A  00C000F8 *   * ..............8 *
 001516A0   * 03C601C4  00330012   0016000D  0010000F *   * .F.D............ *
```

Press **PF3** to return to the Debug Application screen

```
DEBUG APPLICATION ****************** ****************************************
 PROGRAM  ENTRY POINT      DATE        TIME     LENGTH    PCB LIST  TRACE TABLE
TRTMTTDA   00151450     02 02 2001   14 14 59   007BB0    80054054   000B6010

                ID      ADDR          ID      ADDR          ID      ADDR
 TRACE     01 XFERWORK 0015A010   05 PGM WORK 001A8988
 TABLE:    02 WRK STRG 001A8088   06 SSA AREA 001A8730
   *       03 SYS WORK 001A89F0   07 TRGEMPL  001A8CBC
   *       04 APP WORK 001A8848
----------------------------------  ---------------  ------------------------
    FUNCTION                         SECTION  FIELD        MODULE         DISP
  PROGRAM INIT                       Q-100             TRTMTTDA.TRTMTTDA 000130
00151580   * 00000000  0F404040   001515A4  001525A4 *   * .....   ........ *
00151590   * 00152908  00153924   00154932  00155968 *   * ................ *
001515A0   * 00156978  00157160   00158238  00157030 *   * .......-........ *
001515B0   * 00158120  00158088   00157F08  00157EE0 *   * ..............=. *
001515C0   * 00157DB0  00157CD8   00157B48  00157820 *   * .......Q........ *

 TRACE:    PGM INIT Y   FILE BEFORE Y - AFTER Y

COMMAND ==> _____________________________________________ SCROLL ==> PAGE
  AREA:     TRACE ID   ______________  LINE __        DISPLAY LTH  ___
   *        STOP AT ID _______                        UPDATE C/X _
```

Press the TEST GO key to resume program execution, that is, input processing.

In the example below, the TTF displays the Trace DL/I Access Before screen. The Trace Options are still set to stop before and after every data access. Note that:

- The SECTION heading displays **U-100-1**, in which **U** refers to User Exec I/O (**U-100** paragraph), and **1** indicates the first **U-100** paragraph of section **U-100-USER-IO** in the program
- **123456**, the key used to retrieve the data, appears in the SSA LIST field

```
DEBUG APPLICATION ****************** ***************************************
 PROGRAM  ENTRY POINT      DATE        TIME     LENGTH    PCB LIST  TRACE TABLE
TRTMTTDA   00151450     02 02 2001   14 14 59   007BB0    80054054   000B6010

                ID      ADDR          ID      ADDR          ID      ADDR
 TRACE     01 XFERWORK 0015A010   05 PGM WORK 001A8988
 TABLE:    02 WRK STRG 001A8088   06 SSA AREA 001A8730
   *       03 SYS WORK 001A89F0   07 TRGEMPL  001A8CBC
   *       04 APP WORK 001A8848
-----------------------------------  ---------------  ------------------------
    FUNCTION                          SECTION  FIELD        MODULE         DISP
  PROGRAM INIT                        Q-100                 UNKNOWN
0015A010   * 00000000  00000000   00000000  00000000 *   * 123456    ...... *
0015A020   * 00000000  00000000   00000000  00000000 *   * 00010508 ....... *
0015A030   * 00000000  00000000   00000000  00000000 *   * ................ *
0015A040   * 00000000  00000000   00000000  00000000 *   * ................ *
0015A050   * 00000000  00000000   00000000  00000000 *   * ................ *

 TRACE:    PGM INIT Y   FILE BEFORE Y - AFTER Y

COMMAND ==> _____________________________________________ SCROLL ==> PAGE
  AREA:     TRACE ID  _____________  LINE __        DISPLAY LTH  ___
   *        STOP AT ID _______                      UPDATE C/X _
```

Press the TEST GO key to resume program execution and execute the GET HOLD UNIQUE statement.

In the example below, the TTF displays the Trace DL/I Access After screen, which indicates, by spaces under the STATUS heading, that the GHU was successful. Information about the requested employee appears in the I/O AREA.

```
TRACE DL/I ACCESS AFTER  ********** **** ------ PCB INFO ------ **********
 PROGRAM   SECTION    DBD      PCB  FUNC   STATUS  SEGMENT  LEVEL      DLI PARM
 TRTMTTDA  U-100-1  TRGDBDV1   006  GHU            TRGEMPL   01       001841F0

 NO     ADDR    ----- S S A   L I S T -----
  1   801A873C  TRGEMPL *---(TRGEMKEY= 123456)




LENGTH  ADDR     - K E Y   F E E D B A C K   A R E A -     -- CHAR FORMAT ---
   6  000AA5AC   F1F2F3F4 F5F6                            *123456          *
                                                          *                *
LENGTH  ADDR     ----------- I/O   A R E A ------------     -- CHAR FORMAT ---
      001A8CBC   F1F2F3F4 F5F6C2C9    D9C46B40 D3C1D9D9    *123456BIRD, LARR*
      001A8CCC   E8404040 40404040    40404040 404040F5    *Y               5*
      001A8CDC   F8F1F0F2 F4D4F1F1    F3F5F5F5 F1F2F1F2    *81024M1135551212*
      001A8CEC   D6D5C540 C3C5D3E3    C9C340D3 C1D5C540    *ONE CELTIC LANE *
      001A8CFC   40404040 40404040    40C6D9C5 D5C3C840    *         FRENCH *

COMMAND ==> _____________________________________ TRACE BEFORE Y - AFTER Y
OPTIONS ==> UPDATE DLI TRACE _
```

Press the TEST GO key to resume program execution.

In the next example, the TTF displays the Trace DL/I Access Before screen again, indicating REPL under the FUNC heading, as expected, occurring in the **U-100-3** paragraph.

Note: If you decide not to update the record, enter **Y** in the CANCEL CALL field at the bottom of the screen.

```
TRACE DL/I ACCESS BEFORE ********** **** ------ PCB INFO ------ **********
 PROGRAM   SECTION    DBD      PCB  FUNC   STATUS  SEGMENT  LEVEL      DLI PARM
 TRTMTTDA  U-100-3  TRGDBDV1   006  REPL                               001841F0

 NO     ADDR    ----- S S A   L I S T -----




LENGTH  ADDR     - K E Y   F E E D B A C K   A R E A -     -- CHAR FORMAT ---
                                                         *               *
                                                         *               *
LENGTH  ADDR     ---------- I/O   A R E A ------------     -- CHAR FORMAT ---
      001A8CBC   F1F2F3F4 F5F6C2C9    D9C46B40 D3C1D9D9    *123456BIRD, LARR*
      001A8CCC   E8404040 40404040    40404040 404040F5    *Y              5*
      001A8CDC   F8F1F0F2 F4D4F1F1    F3F5F5F5 F1F2F1F2    *81024M1135551212*
      001A8CEC   D6D5C540 C6D3C5C5    E3C3C5D5 E3C5D940    *ONE FLEETCENTER *
      001A8CFC   E6C1E840 40404040    40C2D6E2 E3D6D540    *WAY      BOSTON *

COMMAND ==> ____________________________________ TRACE BEFORE Y - AFTER Y
OPTIONS ==> UPDATE DLI TRACE _                   CANCEL CALL  _
```

Press the TEST GO key to resume program execution.

In the example below, the TTF displays the Trace DL/I Access After screen. Under the STATUS heading, CA Telon displays spaces to indicate that the record was successfully replaced.

```
TRACE DL/I ACCESS AFTER  ************ **** ------ PCB INFO ------ *************
 PROGRAM   SECTION    DBD      PCB  FUNC   STATUS  SEGMENT  LEVEL      DLI PARM
 TRTMTTDA  U-100-3  TRGDBDV1   006  REPL           TRGEMPL   01       001841F0

 NO     ADDR    ----- S S A   L I S T -----




LENGTH  ADDR     - K E Y   F E E D B A C K   A R E A -     -- CHAR FORMAT ---
   6  000AA5AC   F1F2F3F4 F5F6                             *123456          *
                                                          *                *
LENGTH  ADDR     ----------- I/O   A R E A ------------     -- CHAR FORMAT ---
      001A8CBC   F1F2F3F4 F5F6C2C9    D9C46B40 D3C1D9D9    *123456BIRD, LARR*
      001A8CCC   E8404040 40404040    40404040 404040F5    *Y               5*
      001A8CDC   F8F1F0F2 F4D4F1F1    F3F5F5F5 F1F2F1F2    *81024M1135551212*
      001A8CEC   D6D5C540 C6D3C5C5    E3C3C5D5 E3C5D940    *ONE FLEETCENTER *
      001A8CFC   E6C1E840 40404040    40C2D6E2 E3D6D540    *WAY      BOSTON *

COMMAND ==> ____________________________________ TRACE BEFORE Y - AFTER Y
OPTIONS ==> UPDATE DLI TRACE _
```

Press the TEST AID key to access the Debug Application screen.

As requested on the previous TRACE DL/I ACCESS AFTER screen, the TTF displays the Debug Application screen. Since you specified **Y** to stop at Program Init as a Trace Option, the TTF stops at the Debug Screen every time it enters a new program or switches from a program's input processing to the output processing.

As shown below, the TTF has transferred control automatically to the TRTMTTDD program.

```
DEBUG APPLICATION ****************** ****************************************
 PROGRAM  ENTRY POINT      DATE        TIME     LENGTH    PCB LIST  TRACE TABLE
TRTMTTDD   00151508     02 02 2001   14 15 44   006AF8    80054054   000B6010

                ID      ADDR          ID      ADDR          ID      ADDR
 TRACE     01 XFERWORK 0015A010   05 PGM WORK 001A89E8
 TABLE:    02 WRK STRG 001A8088   06 SSA AREA 001A8778
   *       03 SYS WORK 001A8A00   07 TRGEMPL  001A8CCC
   *       04 APP WORK 001A88A8
---------------------------------  ---------------  ------------------------
    FUNCTION                        SECTION  FIELD       MODULE         DISP
  PROGRAM INIT                      Q-100             TRTMTTDD.TRTMTTDD 000000
00151508   * 47F0F028  00C3C5C5   000001E8  00000014 *   * .00..CEE...Y.... *
00151518   * 47F0F001  98CEAC00   001515BE  00000000 *   * .00.............. *
00151528   * 00000000  00000000   90ECD00C  4110F038 *   * ..............0. *
00151538   * 98EFF04C  07FF0000   00151508  00000000 *   * ..0<............. *
00151548   * 00156350  001515B6   00151508  0015349C *   * ...&............. *

 TRACE:    PGM INIT Y   FILE BEFORE Y - AFTER Y

COMMAND ==> ____________________________________________________ SCROLL ==> PAGE
  AREA:     TRACE ID  ________________  LINE __       DISPLAY LTH  ___
   *        STOP AT ID ________                       UPDATE C/X _
```

Press the TEST GO key to execute output processing of the display program.

The Trace DL/I Access Before screen, shown below, indicates that the program is executing a GET UNIQUE in section A-100 using 123456.

```
TRACE DL/I ACCESS BEFORE *********** **** ------ PCB INFO ------ ************
 PROGRAM   SECTION    DBD      PCB  FUNC   STATUS  SEGMENT  LEVEL      DLI PARM
 TRTMTTDD  A-100    TRGDBDV1   006  GU                                 001841C0

 NO     ADDR    ----- S S A   L I S T -----
  1   801A8784  TRGEMPL *---(TRGEMKEY= 123456)




LENGTH  ADDR     - K E Y   F E E D B A C K   A R E A -     -- CHAR FORMAT ---
                                                         *               *
                                                         *               *
LENGTH  ADDR     ---------- I/O   A R E A ------------     -- CHAR FORMAT ---
      001A8CCC   00000000 00000000    00000000 00000000    *...............*
      001A8CDC   00000000 00000000    00000000 00000000    *...............*
      001A8CEC   00000000 00000000    00000000 00000000    *...............*
      001A8CFC   00000000 00000000    00000000 00000000    *...............*
      001A8D0C   00000000 00000000    00000000 00000000    *...............*

COMMAND ==> ____________________________________ TRACE BEFORE Y - AFTER Y
OPTIONS ==> UPDATE DLI TRACE _                  CANCEL CALL  _
```

Press the TEST GO key to resume program execution.

The Trace DL/I Access After screen, shown below, indicates a status of spaces. You can view the updated information in the CHAR FORMAT section of the I/O Area.

```
TRACE DL/I ACCESS AFTER  ************ **** ------ PCB INFO ------ *************
 PROGRAM   SECTION    DBD      PCB  FUNC   STATUS  SEGMENT  LEVEL      DLI PARM
 TRTMTTDD  A-100    TRGDBDV1   006  GU             TRGEMPL   01        001841C0

 NO     ADDR    ----- S S A   L I S T -----
  1   801A8784  TRGEMPL *---(TRGEMKEY= 123456)




LENGTH  ADDR     - K E Y   F E E D B A C K   A R E A -     -- CHAR FORMAT ---
   6  000AA5AC   F1F2F3F4 F5F6                            *123456          *
                                                          *                *
LENGTH  ADDR     ----------- I/O   A R E A ------------     -- CHAR FORMAT ---
      001A8CCC   F1F2F3F4 F5F6C2C9    D9C46B40 D3C1D9D9    *123456BIRD, LARR*
      001A8CDC   E8404040 40404040    40404040 404040F5    *Y               5*
      001A8CEC   F8F1F0F2 F4D4F1F1    F3F5F5F5 F1F2F1F2    *81024M1135551212*
      001A8CFC   D6D5C540 C6D3C5C5    E3C3C5D5 E3C5D940    *ONE FLEETCENTER *
      001A8D0C   E6C1E840 40404040    40C2D6E2 E3D6D540    *WAY      BOSTON *

COMMAND ==> _____________________________________ TRACE BEFORE Y - AFTER Y
OPTIONS ==> UPDATE DLI TRACE _
```

Press the TEST GO key to resume program execution.

The program has now completed output processing and the formatted display appears on the Employee Display screen, shown below.

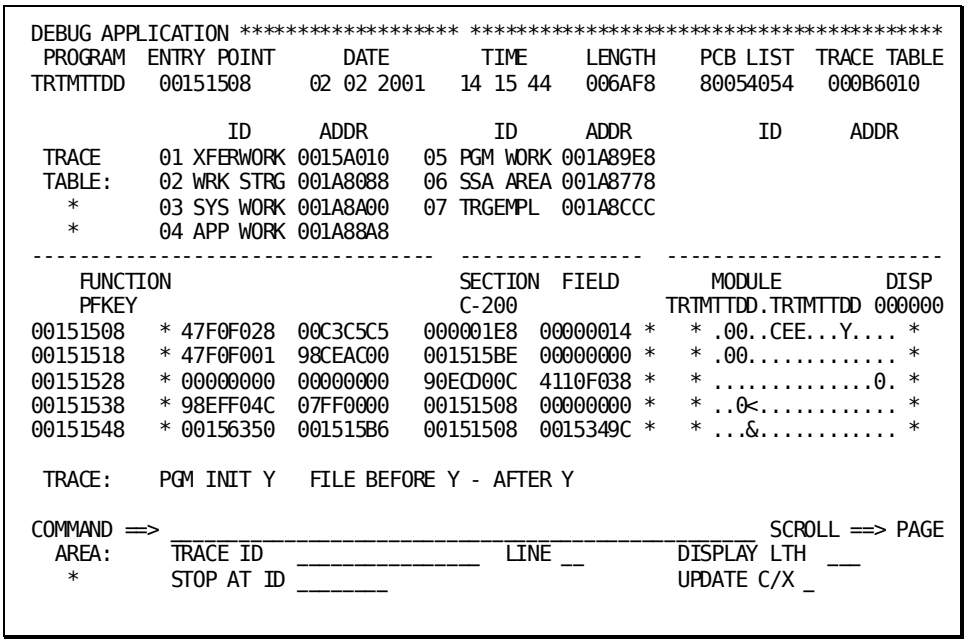
You can now test adding a new employee. To do this, you provide information so that the program can check that the employee number is valid, as illustrated below.

```
05/08/01        T E L O N   S A M P L E   S O L U T I O N
                           EMPLOYEE INQUIRY

          EMPLOYEE ID          123456
          NAME                 BIRD, LARRY

          ADDRESS              ONE FLEETCENTER WAY
                               BOSTON
                               MA   02101   617-555-1212

          DATE OF BIRTH        10/24/58      SEX  MALE
          DATE OF EMPLOYMENT   08/31/84

          HOURS PER WEEK         23.0    DEPARTMENT  BAS
          HOURLY RATE          $20.00
          SALARY              $460.00


  UPDATE EMPLOYEE      _         DELETE EMPLOYEE _
  DISPLAY EMPLOYEE ID  _____     UPDATE TIME CHARGES FOR YEAR __ AND WEEK __


_______________________________________________________________________________
 PFKEYS:  2-HOLD  3-END  4-ENDHOLD  6-MENU  7-PREV  8-NEXT
```

Exit the Test Facility

When you have tested the application to your satisfaction, you can exit the CA Telon Test Facility from any Debug Application screen. To exit:

- Press the TEST AID key to return to the Debug Application screen
- On the Debug screen, perform one of the following:
  - Enter **END** or **=X** on the COMMAND field and press Enter
  - Press **PF3** once to exit to the CA Telon Test Facility Main Menu and press **PF3** again to exit to TSO



Appendix A: Abend Codes

This appendix lists the most common ABEND codes experienced during the testing of CA Telon-generated applications under the CA Telon Test Facility. These include:

- Application program ABENDs
- CA Telon Test Facility ABENDs

Application Program ABENDs

TLND

Reason:

Unexpected file call.

Action:

Check the calls to DL/I. This error frequently occurswhen you omit the ignore parameter on the file's access request statement.

CA Telon Test Facility ABENDs

Computer Associates Development usually needs a dump to solve the following ABENDs.

TLT1

Reason:

Program check in the CA Telon Test Facility.

Action:

Reload the TTF.

TLT4

Reason:

Internal error.

TLTC

Reason:

COBOL thread initialization failed for the test program.

TLTD

Reason:

Error releasing or dequeuing the test program.

TLTL

Reason:

System definition (SDEF) module not in PCT.

Action:

Check install.

TLTO

Reason:

Recursion detected by the TTF.

TLTP

Reason:

More than five pushes of HANDLE in the test application program,or an internal error.

TLTQ

Reason:

Load library has taken extents.

Action:

Reallocate.

TLTR

Reason:

Not used.

TLTS

Reason:

GETMAIN failed. Storage not available for the TTF.

Appendix B: Installation Considerations

This appendix lists and explains CA Telon Test Facility settings relevant to installation.

PF Key Installation Defaults

This section provides the installation defaults for PF Keys.

Test PF Key Assignments

The following PF keys are permanently assigned in the TTF:

PF Key	Command
3/15	END
5/17	RFIND
7/19	BACKWARD
8/20	FORWARD

PF Key Override

The following PF keys are defined at installation time. You can override them on the CA Telon Test Facility Main Menu. The default installation values for these PF keys are:

Default PF Key	Function
PF12.	TEST AID key
PF10 and **PF22**	TEST GO keys

System Definition Module

At installation, CA Telon supplies a System Definition Module TSMASDEF that defines default values for certain options that the CA Telon Test Facility Main Menu displays.

The system definition module TSMASDEF may be defined globally, by application, or by user. To implement it on an application or user level, assemble and link-edit a new module into a separate load library allocated to the TLSYSLIB DD concatenation in the Test Facility CLIST.

You can default the following CA Telon Test Facility Main Menu options from the System Definition Module. Each option is explained below.

TEST AID Key

TSTAID =pa-key/pf-key .— The key used to exit a program being tested and return to the Test Facility Debug screen. Set this to a PF or PA key that is not used globally at your installation. If you use a globally defined key, the results are unpredictable.

The value of this parameter is displayed on the first Test Facility screen where the operator can change it for that particular session.

Values are any PF or PA key; the default is [PF12].

TEST GO Keys

TSTGO= *(gopfkey1, gopfkey2)* — The PF keys the application user uses to issue the GO command from within the Test Facility. You can specify any key except the following pre-assigned Test Facility keys:

- [PF3]/[PF15]
- [PF5]/[PF17]
- [PF7]/[PF19]
- [PF8]/[PF20]

Note: If one of the Test GO keys matches the Test Aid keys, the Test Aid key takes precedence within the Test Facility and the [Enter] key acts as the GO command from the Trace File Access screens.

The defaults are [PF10] and [PF22].

TSO Transfer Work Area Size

XFERSIZ= 32K nK — The default transfer work area size (in 1K increments) to be passed to TSO online programs. Valid values for n are between 3K and 1000K. The default is 32K.

Refresh Working Storage Option

WSREFR= YES NO — Controls whether to REFRESH TSO online programs working storage (except TP buffers) after application screens have been written; values are:

- **YES** — (Default) Yes; this helps ensure that transfer data is not kept in program working storage and prevents this common application error when converting to an IMS/DC environment.
- **NO** — No; do not REFRESH

Appendix C: Commands

All of the available CA Telon Test Facility Commands and their functions are listed in this appendix. You enter all of these commands on the COMMAND field.

Test Facility Commands

This section lists commands alphabetically and within the following categories:

- Address expressions
- Other Test Facility commands
- Commands valid in all TTF screens
- Commands valid on Journal and Queue Test screens
- Commands valid only on Debug screens

Address Expressions

Address expressions appear first in this subject because they require lengthier explanation:

Syntax: Operand1.(>...)({+/-}Operand2)(>...)....

Address expressions consist of certain strings representing an address in core. You can use these expressions on the COMMAND field of either the Debug Application or the Display Full Screen Storage screen to request display of another area of core.

Additionally, address expressions are also used as parameters to other TTF commands, such as AT, to indicate the address for that particular command to process.

Operand1 Values

(module name).

(csect name) (.)

(+/-)....

Display a selected module and optionally a CSECT (subroutine) within that module plus or minus an optional offset. The '.' denotes a module name and if specified alone, results in the display of the beginning (load point) of the current test program.

1 - 32

Display address for entry number 1 through 32 in the Trace Table discussed previously in the Debug Application Screen description.

addr

Display a hexadecimal address in core. Any valid hex string other than the numbers 1-32 is interpreted as a hex address.

Rn

Display the address pointed to by a register, where **n** must be an integer between 0 and 15 inclusive.

PSW

Display the current test program's execution point (Program Status Word).

*****

Use the current address.

***+(N)disp or +N)disp**

Take the current address, add a specified decimal (N) or hexadecimal value to that address and display the resulting address.

***-(N)disp or -(N)disp**

Take the current address, subtract a specified decimal (N) or hexadecimal value from that address and display the resulting address.

***> or >**

Use the first full word of the current display address as a pointer to the address to be displayed.

Operand2-n Values

>

Use the result of the previous operand(s) as a pointer to the address to be displayed.

+/-(N)disp

Add to or subtract from the results of the previous operand(s) a decimal (N) or hexadecimal displacement and display the resulting address.

+/-Rn

Add to or subtract from the results of the previous operand(s) the contents of register **n** and display the resulting address. **n** must be an integer between 0 and 15, inclusive.

Other Test Facility Commands

=X

Syntax: =X

Use this command to exit the CA Telon Test Facility to TSO.

AT

Syntax: AT Address-Expression (stop-id)

Use this command to set a program break point. If you do not supply a stop-id, the TTF uses the last eight characters of the address expression for the stop-id.

AUTOMAP

Syntax: AUTOMAP/AUTOM

Use this command to return to a formatted display of the PL/I Test Program's automatic storage.

BACKWARD

Syntax: BACKWARD/BACK (scroll-value)

Use this command to scroll backward the default or specified scroll-value. The valid scroll-values are HALF, PAGE, and a number of bytes. For screens on which scrolling is active but no scroll field is supplied, a scroll value of PAGE will be used in all scrolling operations.

CANCEL

Syntax: CANCEL/CAN

Use this command to cancel current update. This command is only valid on the Update OS Parameter screen, the Update DL/I Trace screen, and the Update DB2 Trace screen.

DISPLAY

Syntax: DISPLAY/D (STORAGE/S/OSPARM/OSP)

Use this command to transfer to either the Display Full Screen Storage screen or the Update OS Parameter screen.

DOWN

Syntax: DOWN (Scroll-value)

Use this command to scroll down the default or specified scroll-value. The valid scroll-values are HALF, PAGE, and a number of bytes. For screens on which scrolling is active but no scroll field is supplied, a scroll-value of PAGE will be used in all scrolling operations.

END

Syntax: END

Use this command to end current function. From the Application Debug screen, END returns you to either the CA Telon Test Facility Main Menu or the Field Edit Test Menu, depending on the menu from which you originally invoked the Debug Application screen. Most other screens displayed while an Application Test Program is executing will return to the Debug Application screen after you enter the END command. END from the CA Telon Test Facility Main Menu returns you to TSO.

FIND

Syntax: FIND/F search-value

Use this command to find a string in Working Storage and program storage. You must enclose hex strings in single quotes and prefix them with the character X. You must enclose character strings containing spaces or special characters in single quotes. The FIND command searches the program storage for COBOL programs and automatic storage followed by program storage for PL/I. You can use the FIND command on the Debug Application and the Display Full Screen Storage screens only.

FORWARD

Syntax: FORWARD/FO (scroll-value)

Use this command to scroll forward the default or specified scroll-value. The valid scroll-values are HALF, PAGE, and a number of bytes. For screens on which scrolling is active but no scroll field is supplied, a scroll-value of PAGE will be used in all operations.

GO

Syntax: GO/G (nextpgm)

Use this command to continue program execution. If you specify the **NEXTPGM** parameter, execution continues with the **NEXTPGM** specified. If the TTF cannot find the **NEXTPGM**, the TTF returns control to the CA Telon Test Facility Main Menu or Field Edit Test Menu and displays the message PROGRAM DOES NOT EXIST.

ISPF

Syntax: ISPF (option)

Use this command to pass control to PDF. You must invoke the TTF from under PDF for this command to work.

LISTLOAD

Syntax: LISTLOAD/LISTL

Use this command to list current modules loaded in core.

LISTMAP

Syntax: LISTMAP/LISTM (module-name)

Use this command to display the link-edit map for the module that you specify. Omitting the module name parameter results in the display of the link-edit map for the current test program.

LOCATE

Syntax: LOCATE/L search-value (lth)

Use this command to find the string that you specify. The search begins at the current display address. If you do not specify lth, the TTF uses a default length of 10,000 bytes. Locate strings must follow the same syntax defined for the FIND command above. LOCATE occasionally searches less than the specified number of bytes when it encounters fetch-protected core in the search.

MENU

Syntax: MENU

Use this command to return to the CA Telon Test Facility Main Menu screen. The TTF terminates program execution and releases the test program.

OFF

Syntax: OFF (stop-id/address-expression/ALL)

Use this command to remove program break point(s). If you do not supply a parameter, then the TTF removes the current stop-id (if valid) or the current display address. OFF ALL will remove all of the stops currently defined.

PDF

Syntax: PDF (option)

Use this command to pass control to PDF. You must initially invoke the TTF from under PDF for this command to work.

PSW

Syntax: PSW

Use this command to display the test program's current execution point (Program Status Word).

REGS

Syntax: REGS

Use this command to return a formatted display of the current contents of Registers 0 to 15, inclusive.

RELOAD

Syntax: RELOAD

Use this command to reload the current test program and restart program execution at the beginning.

RFIND

Syntax: RFIND/RF

Use this command to repeat last find or locate command. After a successful FIND or LOCATE, RFIND starts its search at the current display address ignoring the previously found value. On a previously unsuccessful FIND command, RFIND starts searching at the beginning of the pre-defined search area as documented previously under the FIND command.

On a previously unsuccessful LOCATE command, RFIND starts its search after the end of the last locate search area. When you issue RFIND after a locate command which has encountered fetch protected core, RFIND does not search any further. You can reset the locate start address by changing the current display address.

SETPSW

Syntax: SETPSW (address-expression)

Use this command to reset the return point of Application Test Program execution. If you do not supply an address expression, the Test Facility uses the current display address. You can only issue SETPSW after an Application Test Program ABEND. The TTF returns you to the debug screen after an ABEND and you enter this command from that screen only.

TRACE

Syntax: TRACE/T DLI/DB2

Use this command to transfer to either the Update DL/I Trace screen or the Update DB2 Trace screen.

TSO

Syntax: TSO command (parms)

Use this command to invoke the TSO command that you specify.

UP

Syntax: UP (scroll-value)

Use this command to scroll up the default or specified scroll value. The valid scroll values are HALF, PAGE, and a number of bytes. For screens on which scrolling is active but no scroll field is supplied, a scroll-value of PAGE will be used in all scrolling operations.

XFERINIT

Syntax: XFERINIT

Use this command to initialize the Transfer Work Area used by the test programs to binary zeros.

Valid on all TTF Screens

- =X
- AUTOMAP
- CANCEL
- END
- GO
- ISPF
- MENU
- OFF
- PDF
- RELOAD
- TRACE

Valid only on Debug Screen

- ADDRESS EXPRESSIONS
- AT
- BACKWARD
- DISPLAY
- DOWN
- EIB
- FIND
- FORWARD
- LISTMAP
- LOCATE
- RFIND
- SETPSW
- UP

Index

U

W